Global Journal of Computer Science and Technology
Graphics & Vision
Volume 12 Issue 12 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performance Analysis of Modified Lifting Based DWT Architecture and FPGA Implementation for Speed and Power

By C.Chandrasekhar & Dr.S.Narayana Reddy
S.V.University, Tirupathi

Abstract - Demand for high speed and low power architecture for DWT computation have led to design of novel algorithms and architecture. In this paper we design, model and implement a hardware efficient, high speed and power efficient DWT architecture based on modified lifting scheme algorithm. The design is interfaced with SIPO and PISO to reduce the number of I/O lines on the FPGA. The design is implemented on Spartan III device and is compared with lifting scheme logic. The proposed design operates at frequency of 280 MHz and consumes power less than 42 mW. The pre-synthesis and post-synthesis results are verified and suitable test vectors are used in verifying the functionality of the design. The design is suitable for real time data processing.

Keywords : *Lifting scheme, low power, high speed, FPGA implementation.*

GJCST-F Classification: *I.4.5*

Strictly as per the compliance and regulations of:

Performance Analysis of Modified Lifting Based DWT Architecture and FPGA Implementation for Speed and Power

C.Chandrasekhar [α] & Dr.S.Narayana Reddy [σ]

Abstract - **Demand for high speed and low power architecture for DWT computation have led to design of novel algorithms and architecture. In this paper we design, model and implement a hardware efficient, high speed and power efficient DWT architecture based on modified lifting scheme algorithm. The design is interfaced with SIPO and PISO to reduce the number of I/O lines on the FPGA. The design is implemented on Spartan III device and is compared with lifting scheme logic. The proposed design operates at frequency of 280 MHz and consumes power less than 42 mW. The pre-synthesis and post-synthesis results are verified and suitable test vectors are used in verifying the functionality of the design. The design is suitable for real time data processing.**

Keywords : Lifting scheme, low power, high speed, FPGA implementation.

I. Introduction

Discrete wavelet transforms (DWT) decomposes image into multiple subbands of low and high frequency components. Encoding of subband components leads to compression of image. DWT along with encoding technique represents image information with less number of bits achieving image compression. Image compression finds application in every discipline such as entertainment, medical, defense, commercial and industrial domains. The core of image compression unit is DWT. Other image processing techniques such as image enhancement, image restoration and image filtering also requires DWT and Inverse DWT for transformations. DWT-IDWT is one of the prominent transformation techniques that are widely used in signal processing and communication applications. DWT-IDWT computes or transforms signal into multiple resolution sub bands [1][2][3][4][5]. DWT is computationally very intensive and consumes power due to large number of mathematical operations. Latency and throughput are other major limitations of DWT as there are multiple levels of hierarchy [6][7][8]. DWT has traditionally been implemented by convolution. Digit serial or parallel representation of input data further decides the architecture complexity. Such an implementation demands a large number of computations and a large storage that are not desirable for either high-speed or low-power applications. Recently, a lifting-based scheme that often requires far fewer computations has been proposed for the DWT. 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 convert the filter implementation into banded matrix multiplications. Since DWT requires intensive computations, several architectural solutions using special purpose parallel processor have been proposed, in order to meet the real time requirement in many applications. The solutions include *parallel filter architecture, SIMD linear array architecture*, *SIMD multigrid architecture, 2-D block based architecture*, and the AWARE's *wavelet transform processor* (WTP) [9][10][11]. Several versions of lifting scheme architecture have been compared and reported in literature. In terms of hardware complexity, the folded architecture in [12] is the simplest and the DSP-based architecture in [13] is the most complex. All other architectures have comparable hardware complexity and primarily differ in the number of registers and multiplexor circuitry. The control complexity of the architecture in [14] is very simple. In contrast, the number of switches, multiplexors and control signals used in the architectures of [15] is quite large. The control complexity of the remaining architectures is moderate. In terms of timing performance, the architectures in [14, 12, 16–18] are all pipelined, with the architectures in [17] having the highest throughput ($1/Tm$). The architecture in [19] has fewer cycles since it is RPA based, but its clock period is higher. The architecture in [17] has the lowest computation delay.

In this paper, we propose, design, model, implement and compare the performances of three different DWT architectures. Section II briefly discusses the Lifting Scheme DWT algorithm for image processing, Section III discusses modified lifting base DWT and Section IV presents the FPGA implementation and compares the results of modified lifting algorithm. Conclusion is presented in Section VI.

II. DWT

The influx of sophisticated technologies in the field of image processing is affiliated with that of

Author α : HOD, Dept.of ECE SVCET, CHITTOOR.
E-mail : Umashekar_2000@yahoo.com
Author σ : Prof & Head in Dept.of ECE, S.V.University, Tirupathi.
E-mail : snreddysvu@yahoo.com

digitization in the computers arena. Image Compression plays an important role of all the Image Processing techniques. The compression techniques are of two types: Lossless and Lossy. The most common image format that uses a lossy compression scheme is JPEG (Joint Photographic Experts Group) format. JPEG 2000 structure is wavelet based compression methodology that provides a number of benefits over the Discrete Cosine Transformation (DCT) compression method, which was used in JPEG format. Wavelet compression converts the image into a series of wavelets that can be stored more efficiently than pixel blocks. The Wavelet compression is accomplished through the use of JPEG 2000 encoder as shown in the figure 1.

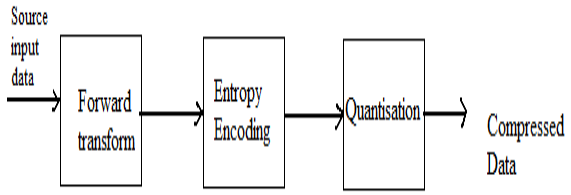

Figure 1 : JPEG 2000 Block Diagram

The problem statement in the present section deals

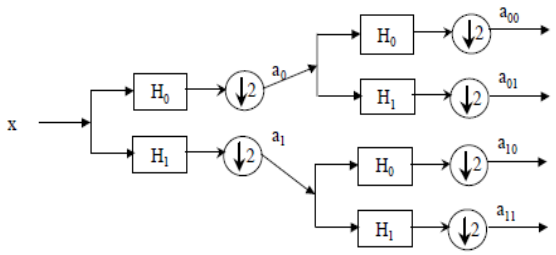

Figure 2 : Two-level DWT decomposition [6]

With the design of the modified two-level DWT architecture for decomposition. The Discrete Wavelet Transform (DWT), which is based on sub-band coding is found top yield a fast computation of Wavelet Transform. It is easy to implement and reduces the computation time and resources required. In DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques [6]. The signal to be analyzed is passed through filters with different cut-off frequencies at different scales as shown in figure 2.

Lifting Scheme:

The Lifting Scheme is a well known method for constructing bi-orthogonal wavelets. The main difference with the classical construction is that it does not rely on the Fourier transform. The lifting scheme is an efficient implementation of a wavelet transform algorithm. It was primarily developed as a method to improve wavelet transform, and then it was extended to a generic method to create so-called second-generation wavelets. Second-generation wavelets are much more flexible and powerful than the first generation wavelets. The lifting scheme is an implementation of the filtering operations at each level [6]. The figure 3 represents the classical and lifting based implementations of DWT.

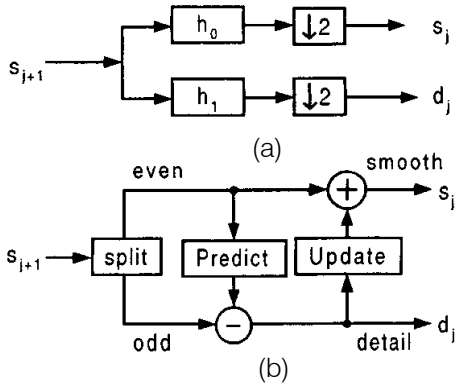

Figure 3 : a) Classical Implementation, b) Lifting scheme based DWT [6]

Lifting Scheme consists of three steps: SPLIT, PREDICT and UPDATE, as shown in the figure 3 (b).

- **SPLIT:** In this step, the data is divided into ODD and EVEN elements.
- **PREDICT:** The PREDICT step uses a function that approximates the data set. The differences between the approximation and the actual data, replace the odd elements of the data set. The even elements are left unchanged and become the input for the next step in the transform. The PREDICT step, where the odd value is "predicted" from the even value is described by the equation [6].

$$Odd_{j+1,\,i} = odd_{j,\,i} - P\,(even_{j,\,i})$$

- **UPDATE:** The UPDATE step replaces the even elements with an average. These results in a smoother input for the next step of the wavelet transform. The odd elements also represent an approximation of the original data set, which allows filters to be constructed. The UPDATE phase follows the PREDICT phase. The original values of the odd elements have been overwritten by the difference between the odd element and its even "predictor". So in calculating an average the UPDATE phase must operate on the differences that are stored in the odd elements [6]:

$$Even_{j+1,\,i} = even_{j,\,i} + \boldsymbol{U}\ (odd_{j+1,\,i})$$

The equations for the lifting based implementation of the bi-orthogonal wavelet are:

$$\text{Predict P1: } d_i^1 = \alpha\,(x_{2i} + x_{2i+2}) + x_{2i+1}$$
$$\text{Update U1: } a_i^1 = \beta\,(d_i^1 + d_{i-1}^1) + x_{2i}$$
$$\text{Predict P2: } d_i^2 = \gamma\,(a_{i1} + a_{i+1}) + x_{2i+1}$$
$$\text{Update U2: } a_i^2 = \delta\,(d_i^2 + d_{i-1}^2) + a_i^1$$

Scale G1: $a_i = \zeta\, a_i^2$
Scale G1: $d_i = d_i^2/\zeta$

The figure 4 shows the lifting scheme architecture to realise the equations shown above.

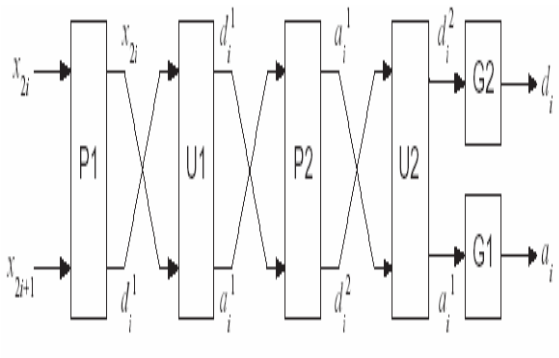

Figure 4 : Lifting Scheme Architecture

The input data x is first split into even and odd samples and each of the samples are taken through predict and update stages as per the architecture shown above. As the data moves from first stage to the last stage, data switching occurs at the input and output of every stage. Every stage consists of multipliers and adders. For the given set of Predict and Update stages, assuming the value of i = 0, the equation can be finalized.

III. Modified Lifting Scheme

By re-arranging all the values and the constant co-efficient, the final equation can be derived.

$$ai = (3 * \underline{\gamma.\beta.\delta.\zeta + \delta.\zeta + \beta.\zeta}) [\alpha(x_0 + x_2) + x_1 + \alpha(x_0 + x_{-2}) + x_{-1}] + \underline{\zeta.\delta.\beta.\gamma}[\alpha(x_2 + x_4) + x_3 + \alpha(x_{-2} + x_{-4}) + x_{-3}] + \underline{\zeta.\delta.\gamma}(x_0 + x_2 + x_0 + x_{-2}) + \underline{\zeta} * x_0$$

$$di = 1/\zeta [(2 * \underline{\gamma.\beta} + 1)\{\alpha(x_0 + x_2) + x_1\} + \underline{\gamma.\beta}\{\alpha(x_0 + x_{-2}) + x_{-1} + \alpha(x_2 + x_4) + x_3\} + \underline{\gamma}(x_0 + x_2)].$$

Being a dedicated DWT core for JPEG 2000, the filter coefficients are fixed. The filter coefficients are: $\alpha = 1.586134342$, $\beta = 0.05298011854$, $\gamma = 0.8829110762$, $\delta = 0.4435068522$, $\zeta = 1.149604398$. By substituting the above values in the modified equation, the coefficient values obtained then are also decimals, by multiplying them with constants they form integers as: 1 * 32 = 57, 2 * 256 = 6, 3 * 64 = 30, 4 * 32 = 35, 5 * 256 = 12, 6 * 32 = 26, 7 * 32 = 50.

Thus the above integers are the values of the underlined coefficients in above equations. From the equations it is observed that there are common lifting coefficients to compute ai and di coefficients and there are input terms. The architecture realised by the above equations considering the constant coefficients is shown in the figure 5.

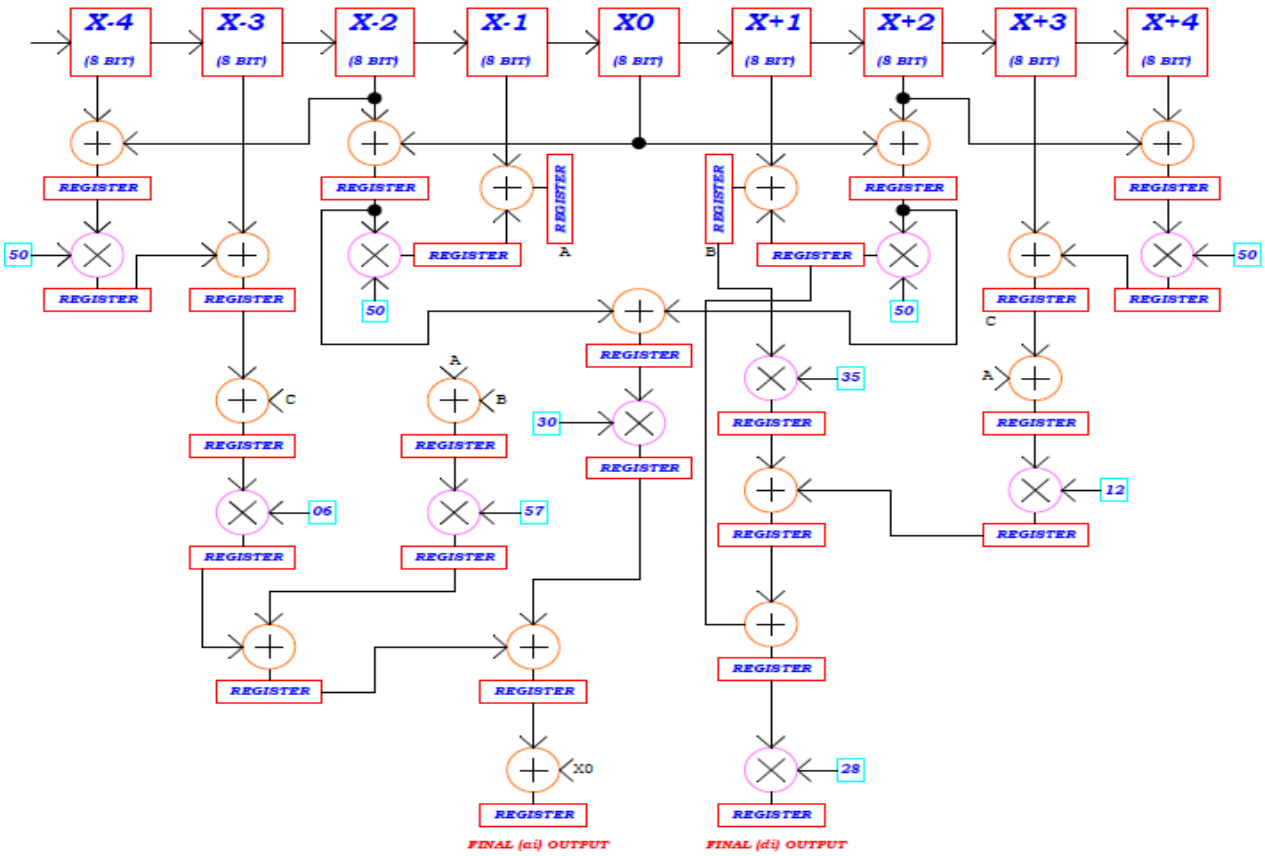

Figure 5 : Modified Lifting Scheme Architecture for DWT

The FPGA implementation of the modified lifting based DWT is designed based on the following:

- The input data X should be of 8 bit signed data.
- Output should be of 16 bit signed representation (including a_i and d_i).
- The lifting coefficients should be of 8 bit signed representation.
- The architecture should work on the streaming input sequence (Serial input).
- The intermediate outputs should be stored in a memory.
- The input data flows into the architecture through one input line and the output should be read out through one output line.

Thus by considering the above design specifications the architecture shown in the figure 5 is designed as per the requirements.

- The blocks from X -4 to X +4 resemble the input 9 samples designed in form of SIPO, each of 8-bit signed representation (serial in parallel out).
- Here the input stream is given through the single input line to 9 SIPOs. The outputs of those are taken in parallel to perform the addition and multiplication operations.
- The addition and multiplication operations are of 8 bit signed operators.
- The intermediate results of these addition and multiplication operations are stored in registers than preferring memory, as the data can be stored in registers with ease and in random, but in memory the storage (write operation in particular) should be done in orderly fashion.
- These intermediate registers are of PIPO structure and of 8 bit signed representation.
- Though final outputs ai and di are single bit, those are stored in the registers of PISO structure as the output should be taken for 8 bits.

The inputs can be 8-bit signed at any point of time. But the outputs should not be a signed number and can be more than 8-bits as every time the adders create an extra bit and the multipliers create more than one bit of data. That might be the major cause for the failure of the hardware; the architecture might not work properly. In order to minimize the error, suitable modifications are carried out.

Modifications to minimize the Errors:

The few possible modifications that can be done for the calculations which can minimize the errors are:

1. An adder performs the addition of two 8-bit numbers and gives the result as a 9-bit number. Instead of a 9-bit number the LSB is discarded. As the Least Significant Bit is discarded the value of the number might not change drastically and the output still is an 8-bit data which is used for further operations.
2. The multiplier performs multiplication of one 8-bit number and the other is coefficient numbers. For each multiplication the hardware will be different so the final architecture requires a lot of multipliers which are of different width and again gives different output values.
3. The lifting co-efficient which should be of 8-bit signed number goes in decimal numbers like **0.458** so that the computation will become very difficult. For multiplying this number multiplier takes more time to compute and the final output would be a decimal as **57.35.**

The lifting coefficient is multiplied with an integer as **57** so that representing it might be much easier. These coefficients should be multiplied such that the final values should be obtained as 8-bit signed number and it should not have any decimal value as **57.000.** This can be achieved by taking only the positive values and discarding all other decimal point values (e.g.XY.xy). Thus the values of all the lifting coefficients have integer without any decimal values so that the calculations can be much easier. From the architectural calculations, the values of ai and di are **65** and **39** respectively, and match with theoretical calculations.

By comparing those values we can come to know that:

1. The architecturally calculated values are of 8-bit singed representation while theoretically calculated are unsigned.
2. The architectural values do not have any decimal values.
3. The architectural values do not exceed more than 8-bit.
4. The intermediate calculations will be always 8-bit and signed instead of 9 or more bits.
5. The outputs of the adder in architectural calculations are 8-bit by discarding the LSB than having 9-bits which will be continued to increase for next level of addition.

Estimation of Power, Area and Delay of Sub-Blocks of Architecture:

The main sub-blocks of the modified lifting scheme architecture are:

- ✓ Adders
- ✓ Multipliers (Constant Coefficient-IP Cores)
- ✓ Registers

The table 1 represent the estimation of Power, Delay and Area of these sub blocks.

Table 1 : Estimation of Power, Delay and Area of sub blocks

Sl No	Sub Block	Power (W)	Area(no of slices,utilization of slice%)	Delay(nS)
1)	Adder	0.142	5(1%)	3.369
2)	Multiplier(50)	0.007	9 (1%)	1.999
3)	Multiplier(30)	0.037	7(1%)	0.881
4)	Multiplier(12)	0.037	6(1%)	0.881
5)	Multiplier(35)	0.037	13(1%)	0.881
6)	Multiplier(57)	0.037	11(1%)	0.881
7)	Multiplier(6)	0.037	6(0%)	0.881
8)	Multiplier(26)	0.037	7(1%)	0.881
9)	PISO	0.011	5(1%)	1.216
10)	SIPO	0.037	36(1%)	0.915
11)	PIPO	0.037	1(0%)	0.916

The table 1 represents the estimation of Power, Delay and Area of the Sub-blocks. Here every block is of 8 bit signed representation from +127 to -128. As observed from the table there are one type of adder, multipliers of 7 types of constant coefficient type to reduce the complexity of multiplication the IP cores in XILINX are used. Of all the sub-blocks the Adder has the highest delay and the highest utilisation of the resources. Thus by instantiating these sub-blocks the area utilised by the DWT architecture is 12% and the delay is 3.313ns. From the table 1 the delays of individual blocks are known. Almost all work at different clock frequencies, as the delay mentioned in the table is the minimum period of the design clock. But the whole or the top level design should work at one clock frequency, thus the concept of synchronising the clocks arise. The clock frequency of top level architecture should synchronise with the sub modules, in general the problem of Synchronisation is addressed by any of these below:

- Increase system clock period (usually not feasible).
- Decrease tcomb (use no combinational logic).
- Decrease tsu (use fast flip-flops)
- Increase synchroniser clock period.

The figure 6 represents the clock synchroniser.

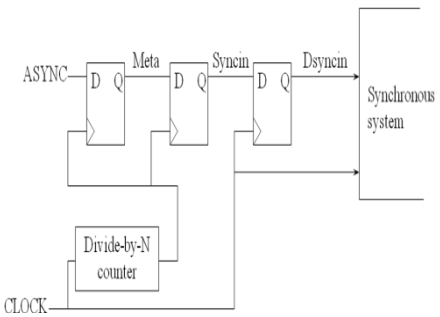

Figure 6 : Clock Synchroniser

In applying the same for the implementation of the present top-level design architecture, the DCM is used. The Divide-by-N counter block in the figure 7 can be replicated by using DCM. The figure 2.8 represents how to configure the DCM.

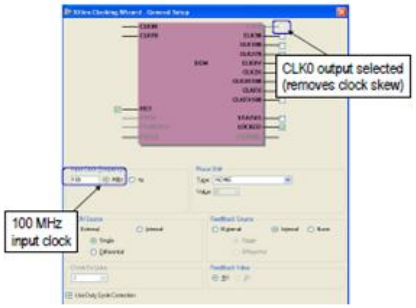

Figure 7 : Configuring DCM [2]

Thus by configuring the DCM in the frequency mode the tool generates the instantiation template and thus that instantiation template is used in the design to make the design run on same clock. The operating frequency of the present design runs at 280MHz.

Observations:

- The equation of the lifting scheme for two-level DWT is simplified based on the basic equations mentioned.
- The simplified equation is made into an architecture such that both the ai and di is implemented using the same architecture.
- The mathematical and the architectural computation of the equation are computed and compared, and observed that the architectural computations are the modified version of the mathematical, where the discarding of the LSBs result to scaling down of the original values.
- The power, area and the delay of the sub-blocks are observed and noticed that the Adder takes the maximum delay i.e. 3.39ns and maximum utilisation of resources i.e. 13%, and registers SIPO and PIPO takes the least delay and least utilisation of resources i.e. 1%

IV. HDL Modeling and FPGA Implementation

The top level module or block of the DWT architecture is shown in the figure 8. The figure explains the input and output ports. The input ports are **clk, en, piso_load, rst and ser_in** and the output ports are **ai** and **di**. The input 9 samples each of 8 bit signed data is entered into the design through the **ser_in** input. The **rst** signal is used to reset the design when the signal is high. When the **en** signal is high, loading of the input data in all the 9- 8 bit registers for 280 clock cycles is done. The **piso_load** signal is used to take the output at **ai** and **di,** and this signal is kept high for 8 clock cycles

as the 8 bit is to be taken out through the single line. The HDL models of the sub-block can be understood from the internal hardware of the RTL schematic shown in the figure 9. The figure 9 represents the schematic of the DWT architecture where all the sub blocks can be viewed. Thus the sub blocks are modelled in such a way that the multipliers used are the IP cores from the XILINX library, and the adder that is designed for 8 bit signed addition is instantiated wherever necessary. The simulation of the top level module is shown in the figure 10 where the intermediate signals gives the performance of the sub blocks in the total simulation.

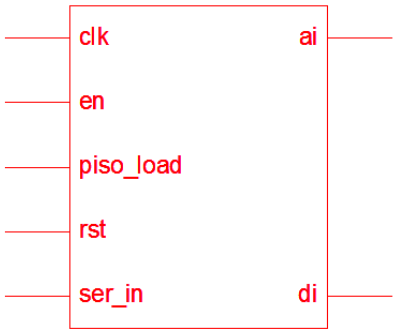

Figure 8 : Top-level DWT architecture

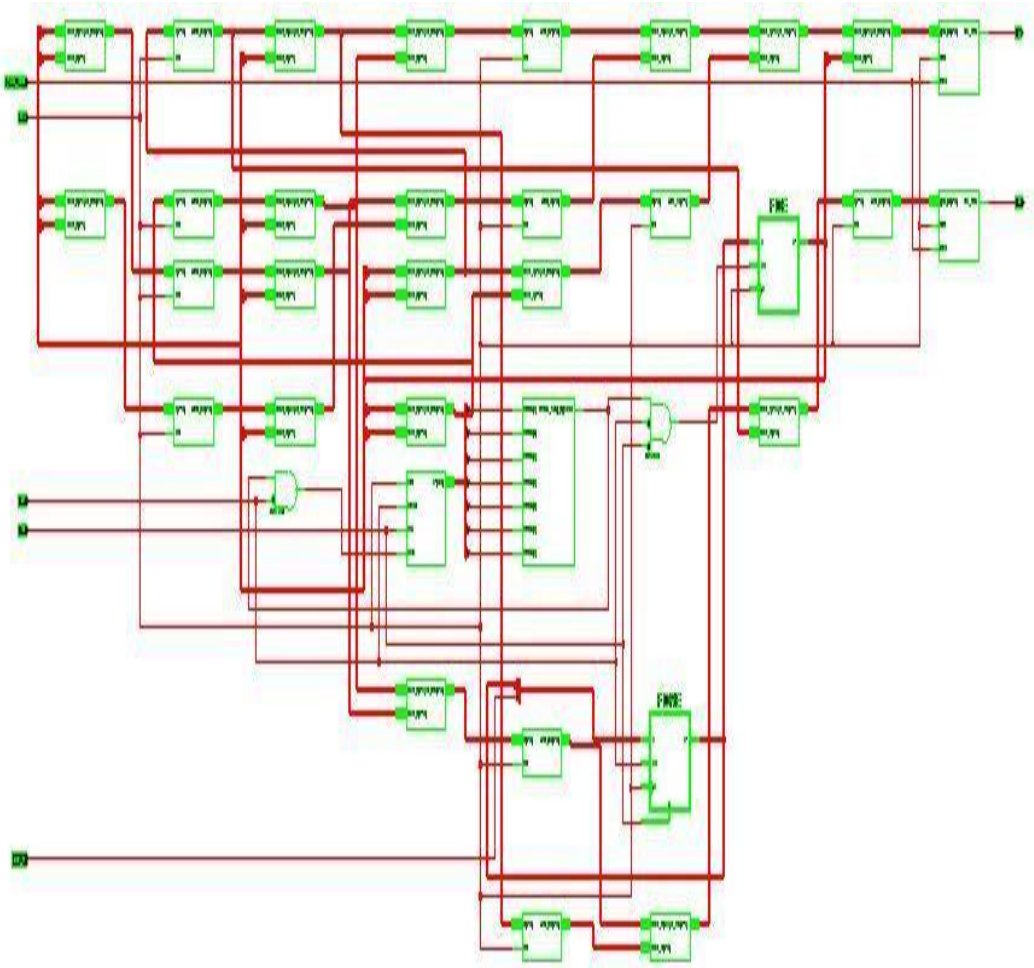

Figure 9 : RTL schematic of DWT showing sub-block

The figure **explains the integration of the sub blocks** in the main top level architecture. Initially the sub blocks are designed by considering the DWT equation, the multiplier used in the design is a **constant coefficient multiplier** as it is faster than any other for the application required. For the present design, the constant coefficient multipliers are taken as a **IP core** from the XILINX library for different coefficients. The **adder** is 8-bit signed operator designed or modelled in the HDL and instantiated where it is necessary. The registers that are used in the design covers all the types SIPO, PIPO, PISO. SIPO at the initial stage while giving the inputs, PIPO while performing the operations intermediately, and the PISO at the output stage to take the outputs serially i.e. one bit for 8 clocks, as the required is two outputs of 8 bits taking serially. From the figure 9, the top level ports are shown; the serial input data is given in a random way. This is loaded inthe registers (SIPO), when enable signal is high, after 72 clock cycles the enable is made low, and for four clock cycles the operation is performed and the output is taken when the piso_load signal is high for 8 clock cycles as the output

is taken for 8 bit. Thus the same procedure follows for 8 (load)+ 4 (operation) + 8 (output) = 22 clocks. To program a single device using iMPACT, all needed is a bitstream file. To program several devices in a daisy chain configuration, or to program devices using a PROM, iMPACT is used to create a PROM file. iMPACT accepts any number of bitstream and creates one or more PROM files containing one or more daisy chain configurations.

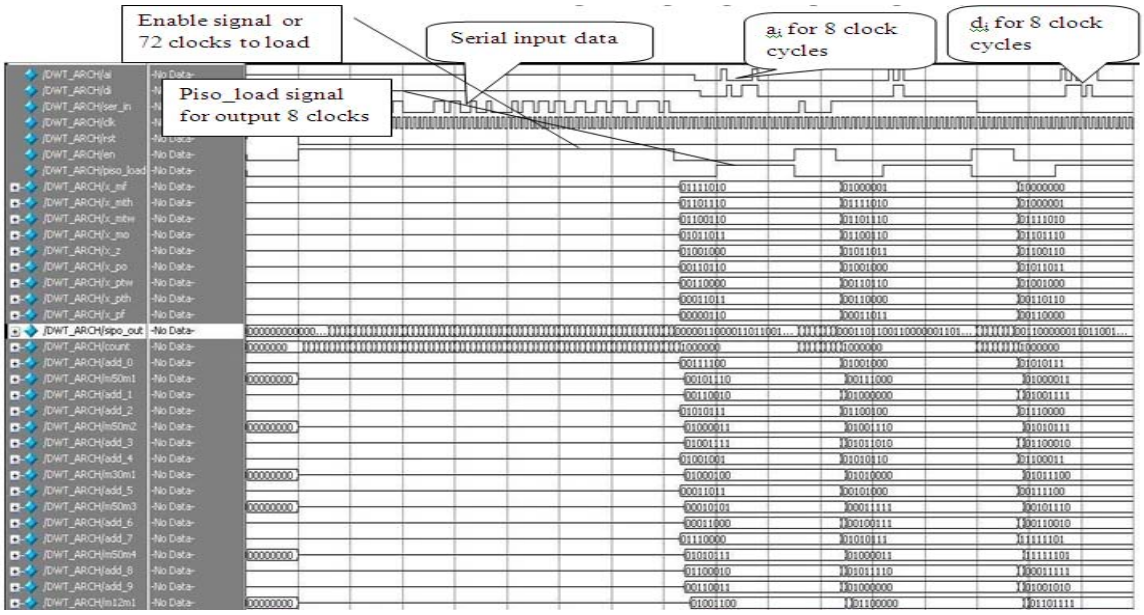

Figure 10 : Simulation results for the DWT architecture

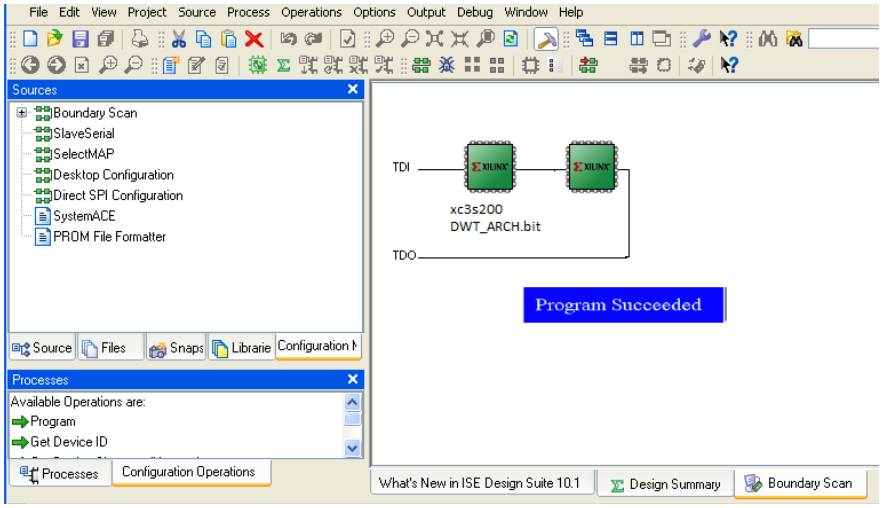

Figure 11 : Program downloaded into FPGA

Comparison of Post and Pre- Synthesis results:

The Post-synthesis result is that obtained after the place and route process has done. Here the delays of the LUTs and also the interconnection delay are added. But the Pre-Synthesis is nothing but the behavioral simulation. The comparison between the Post and Pre Synthesis results is shown in the figure 12 and 13.

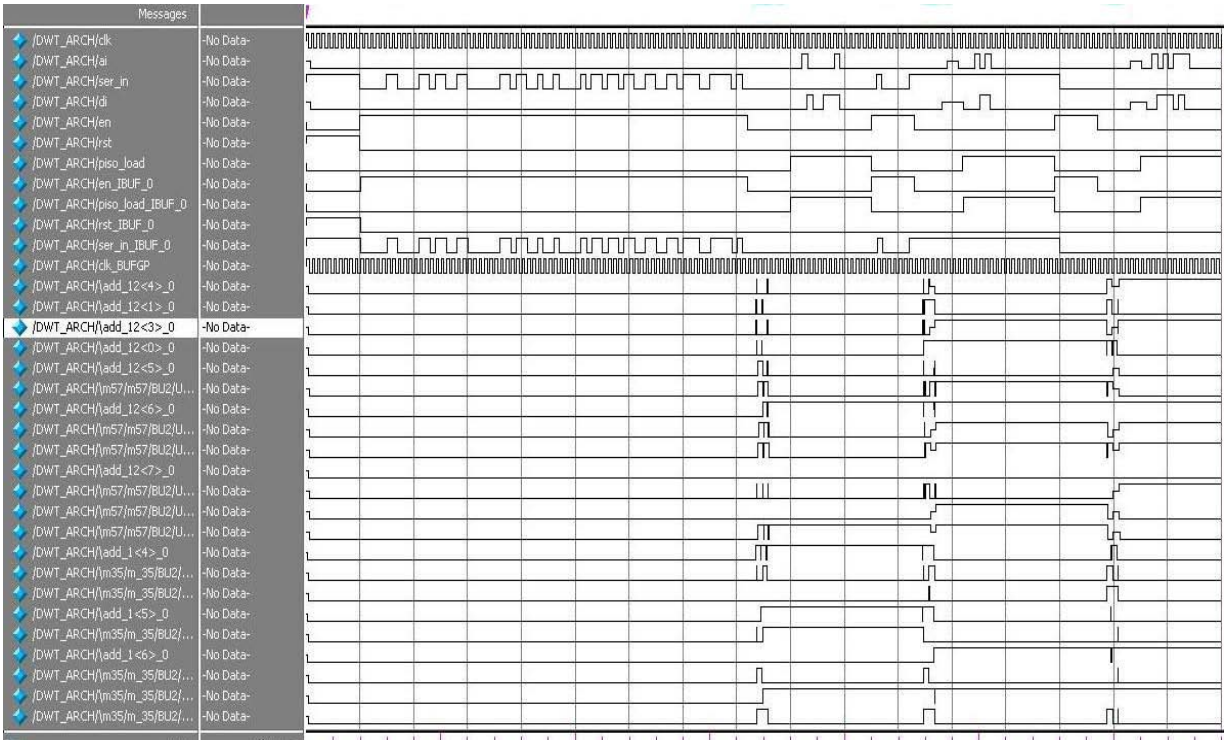

Figure 12 : Post-Synthesis simulation

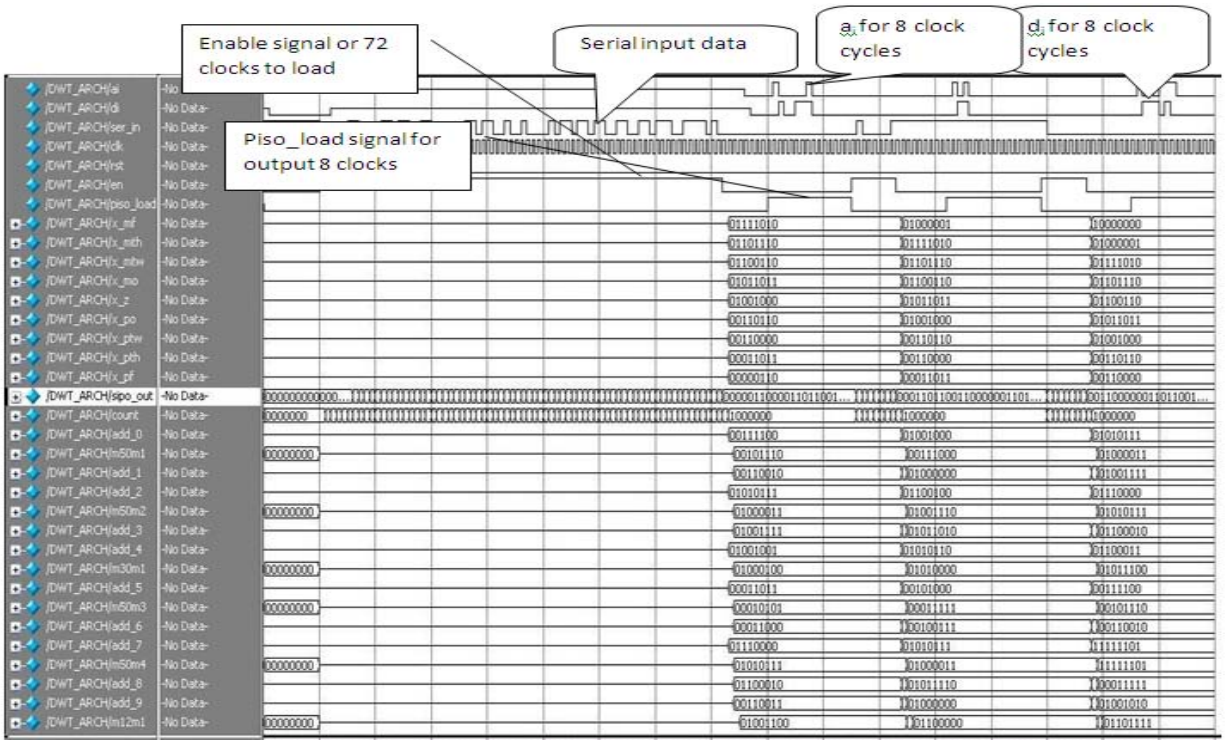

Figure 13 : Pre-Synthesis Simulation

Logic Utilization

Number of Slice Flip Flops: 247 out of 1,536
Number of 4 input LUTs: 295 out of 1,536

Logic Distribution

Number of occupied Slices: 245 out of 768

Number of Slices containing only related logic: 245 out of 245

Number of Slices containing unrelated logic: 0 out of 245

Total Number of 4 input LUTs: 314 out of 1,536

Number used as logic: 295

Number used as a route-thru: 19
Number of bonded IOBs: 6 out of 140
Number of GCLKs: 1 out of 4
Number of GCLKIOBs: 1 out of 4

Place and Route

Placement and routing is performed by the PAR program. *Place and route* is the most important and time consuming step of the implementation. It defines how device resources are located and interconnected inside an FPGA. Placement is even more important than routing, because bad placement would make good routing impossible. In order to provide possibility for FPGA designers to tweak placement, PAR has a "starting cost table" option. PAR accounts for timing constraints set up by the FPGA designer. If at least one constraint can't be met, PAR returns an error. The output of the PAR program is also stored in the NCD format. The device utilization summary of the architecture is given below.

Device Utilization Summary

- Number of GCLKs: 1 out of 4 25%
- Number of External GCLKIOBs: 1 out of 4 25%
- Number of LOCed GCLKIOBs : 0 out of 1 0%
- Number of External IOBs: 6 out of 140 4%
- Number of LOCed IOBs: 0 out of 6 0%
- Number of SLICEs: 245 out of 768 31%

Table 2 : Performance comparison of DWT architecture

Parameters	DWT [9]	Lifting DWT [18][19]	Modified Lifting DWT
No of Slices	432 out of 1536	358 out of 1536	247 out of 1536
No of gates	37K	27K	15K
Clock Speed	72 MHZ	156 MHz	268 MHz
Power dissipation	81 mW	67 mW	42 mW

V. Conclusion

In this work a modified lifting based DWT architecture is proposed, designed, modeled and verified. The design is modeled using HDL and is implemented on FPGA. The interfaces requried for data processing are also designed and is used to synchronize the data transfer operation. The HDL models and simulation of the sub blocks have been done to model the top-level design architecture. The test-bench to verify the functionality and performance of the sub modules and the top level architecture have been done. Implemented the design on FPGA and verified and debugged through the Chip-Scope. The Pre and Post Synthesis have been done and compared. The design can be further optimized for video signal processing.

References Références Referencias

1. N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
2. C. Diou, L. Torres, and M. Robert, "A wavelet core for video processing," presented at the IEEE Int. Conf. Image Process., Sept. 2000.
3. N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proc. IEEE*, vol. 81, pp. 1385–1422, Oct. 1993.
4. B. Zovko-Cihlar, S. Grgic, and D. Modric, "Coding techniques in multimedia communications," in *Proc. 2nd Int. Workshop Image and Signal Processing, IWISP'95*, Budapest, Hungary, 1995, pp. 24–32.
5. *Digital Compression and Coding of Continuous Tone Still Images*, ISO/IEC IS 10918, 1991.
6. I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM, 1992.
7. S. Mallat, "A theory of multiresolutio n signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.
8. M. Nagabushanam, Cyril Prasanna Raj P, S. Ramachandran, Design and Implementation of Parallel and Pipelined Distributive Arithmetic Based Discrete Wavelet Transform IP Core, European Journal of Scientific Research ISSN 1450-216X Vol.35 No.3 (2009), pp.378-392.
9. M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 305–316, May 1995.
10. J. S. Fridman and E. S. Manolakos, "Discrete wavelet transform: Data dependence analysis and synthesis of distributed memory and control array architectures," *IEEE Trans. Signal Processing*, vol. 45, pp. 1291–1308, May 1997.
11. T. Acharya, "A high speed systolic architecture for discrete wavelet transforms," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, 1997, pp. 669–673.
12. C.J Lian, K.F. Chen, H.H. Chen, and L.G. Chen, "Lifting Based Discrete Wavelet Transform Architecture for JPEG2000," in *IEEE International Symposium on Circuits and Systems*, Sydney, Australia, 2001, pp. 445–448.
13. M. Martina, G. Masera, G. Piccinini, and M. Zamboni, "Novel JPEG 2000 Compliant DWT and IWT VLSI Implementations," *Journal of VLSI Signal Processing*, vol. 34, 2003, pp. 137– 153.
14. C.C. Liu,Y.H. Shiau, and J.M. Jou, "Design and Implementation of a Progressive Image Coding

Chip Based on the Lifted Wavelet Transform," in *Proc. of the 11th VLSI Design/CAD Symposium*, Taiwan, 2000.

15. H. Liao, M.K. Mandal, and B.F. Cockburn, "Efficient Architectures for 1-D and 2-D Lifting-BasedWavelet Transform," *IEEE Transactions on Signal Processing*, vol. 52, no. 5, 2004, pp. 1315–1326.
16. W.H. Chang, Y.S. Lee,W.S. Peng, and C.Y. Lee, "A Line-Based, Memory Efficient and Programmable Architecture for 2D DWT Using Lifting Scheme," in *IEEE International Symposium on Circuits and Systems*, Sydney, Australia, 2001, pp. 330–333.
17. C.T. Huang, P.C. Tseng, and L.G. Chen, "Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform," in *IEEE Transactions on Signal Processing*, 2004, pp. 1080–1089.
18. 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.
19. 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.