Journal of Theoretical and Applied Information Technology
31st December 2020. Vol.98. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HARDWARE ACCELERATOR FOR COMPUTING DTCWT SUB BANDS USING SPLIT LIFTING ALGORITHM

[1]SUNITHA . P.H , [2]SREERAMA REDDY G. M, [3]CYRIL PRASANNA RAJ. P

[1]Associate Professor, Department of ECE, M.S. Engineering College, Bangalore, India.

[2]Principal, C Byregowda Institute of Technology, Kolar, India.

[3] Professor and Dean R & D, Department of ECE, M.S. Engineering College, Bangalore, India.

[1]sushreeanju@gmail.com, [2]sreeramareddy99@gmail.com, [3]cyrilyahoo@gmail.com

ABSTRACT

DTCWT based image processing is gaining popularity for its advantages such as shift invariance and additional directional features compared with DWT. Computation complexity of DTCWT have limited its use for real time image processing applications. Hardware accelerators for image processing algorithms implemented on FPGA platform have demonstrated improvement in computation speed. In this work, hardware accelerators for computing DTCWT based on split lifting scheme algorithm is designed and implemented on FPGA. The number of arithmetic operations and complexity in performing multiplication is reduced by multiplierless operations and reuse logic. The proposed design is implemented on FPGA and is demonstrated to operate at maximum frequency of 333 MHz and power dissipation is limited to 56 W. The DTCWT computation is reconfigurable to perform both forward and inverse transforms.

Keywords: *Hardware Accelerator, DTCWT, Lifting Scheme, Multiplierless Logic, Reuse Logic .*

1. INTRODUCTION

Over the last three decades Discrete Wavelet Transform (DWT) has been widely used for multiresolution image and signal processing applications including image registration, compression, fusion, classification and orthogonal frequency division multiplexing. The popularity of wavelet based algorithms is further to grow in future with significant research progress in design of filter banks for wavelet transforms. Lack of directionality from wavelet sub bands, poor support for shift invariant property and lack of phase information are the major limitations of wavelet based filtering. Dual Tree Complex Wavelet Transform (DTCWT) has been carried out to overcome the limitations of DWT and studies on filter bank design are also being worked out by many of the researchers [1]. Computation complexity of DTCWT algorithm in decomposing input images into multiple levels have constrained use of DTCWT for real time image processing [2]. Image processing algorithms process input image which is of 2D or more and processing is carried out by grouping image into sub images of smaller size and hence requires parallel processing operations so that all the sub images can be processed simultaneously. As the volume of data that is acquired from imaging devices increases the amount of data that need to be processed also increases and processing huge volume of data in real time requires multicore platforms that can process data using parallel processing algorithms. In CPU or GPU based processing platform the concept of fetch-decode-execute operations delay the process of executing introducing latency. Fetching of data from the cache memory may miss out and frames could be dropped during data access when the processing speeds are fixed. In FPGA based image data processing predictability is very high with frames per second processing meeting the requirements of real time image processing specifications. Power consumption is another factor that is advantageous in FPGA compared with CPU or GPU. The work carried out by Fowers et al. in 2012 [3] have demonstrated that image processing algorithm implemented on FPGA required only 20 Watts of power as compared with the same algorithm implemented on CPU and GPU consuming 130 Watts and 145 Watts respectively [4]. In CPU and GPU dynamic memory allocation concept is advantageous which is a limitation in FPGA is because of limited availability of on-chip memory. For portable medical devices power dissipation by image processing module need to limited to less than 5 Watts for durable operation of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the device. Xilinx Zedboards cannot accommodate more than 4 images of size 1024 x 768 as compared with Kintex 7 FPGA [5]. The processing power and the features available on Zedboard need to be utilized and customization in architecture is required so as to meet the cost requirements. Next generation FPGAs such as Ultrascale+ devices are designed to support higher density memory options supporting complex image processing applications [6]. Even with increasing the memory storage on FPGAs as claimed by new memory technologies such as Hybrid Memory Cube (HMC) [7] and High-Bandwidth memory (HBM) [8] will still not support the memory requirements of complex image processing algorithms. However, FPGAs have large number of distributed memory units that can be used as local memories. Utilizing the local memory units on FPGA that are distributed uniformly across the FPGA architecture is the right way ahead for implementing complex image processing applications which requires proper architecture design and use of coding models. Ioannis Stratakos et al. in 2019 [9] have presented hardware accelerators for image registration on FPGA platform with System on Chip approach. Optimizing process is carried out using Downhill simplex method and Powells direction method and similarity measure is carried out using correlation coefficient and mutual information techniques. Robert Stewart et al. in 2018 [10] have presented discussion on Rathlin Image Processing Language (RIPL) a high level image processing language specifically for modelling algorithms on FPGAs. The concept of map and zip with inspired by stream-based functional programming languages and libraries are considered as primitives and called as skeletons in RIPL. The skeletons capture the basic image processing arithmetic operations, 1D and 2D filtering operation, transformation operations and image reduction operation. Hardware pipelines are generated in RIPL to improve latency of image processing algorithms. The macro blocks and libraries in RIPL do not support DTCWT architecture and it is required to develop the architecture considering basic sub systems. Hardware accelerators implemented on FPGA have been demonstrated to achieve more than 36X of speed improvement for the modules such as affine transformation, similarity measure, DCT, FFT etc. Very few literatures have been reported on hardware accelerators for DTCWT implementation for image processing applications. In this paper, detailed discussion on design of high speed architectures for performing arithmetic operations and data movement operations in computing complex wavelet sub bands is presented.

2. LIFTING SCHEME

One of the simplest methods for wavelet filter bank implementation is using lifting scheme that was first introduced by Sweldens in 1995 [11]. The work carried out by Daubechies and Sweldens in 1998 [12] have implemented lifting scheme for FIR filters. The lifting scheme coefficients are determined using the process of factorization of Polyphase matrix. Implementing filter bank structure using lifting scheme reduces the computation complexity on hardware platforms. Both inverse and forward transforms can be carried out by lifting scheme leading to perfect reconstruction.

2.1 Lifting Scheme For DTCWT

DTCWT filters as discussed in previous chapters for the analysis filter bank structure are represented as $\{h_0(n), h_1(n)\}$ and $\{g_0(n), g_1(n)\}$ for the real and imaginary tree respectively. The filter banks for the synthesis filters are $\{f_0(n), f_1(0)\}$ and $\{p_0(n), p_1(n)\}$ representing real and imaginary tree filter structure. Polyphase notation for these filters are expressed as in Eq. (1),

$$H_0(z) = H_{00}(z^2) + z^{-1} H_{01}(z^2)$$
$$H_1(z) = H_{10}(z^2) + z^{-1} H_{11}(z^2)$$
$$G_0(z) = G_{00}(z^2) + z^{-1} G_{01}(z^2)$$
$$G_1(z) = G_{10}(z^2) + z^{-1} G_{11}(z^2) \quad (1)$$

Expressing filters using Polyphase matrices of $\{H_0(z), H_1(z)\}$ and $\{G_0(z), G_1(z)\}$ in terms of $H_p(z)$ and $G_p(z)$ respectively and these parameters are represented in terms of even and odd phases of filters as in Eq. (2) for the filter $G_p(z)$ and similarly can be expressed for $H_p(z)$.

$$G_p(z) = \begin{bmatrix} G_{00}(z) & G_{01}(z) \\ G_{10}(z) & G_{11}(z) \end{bmatrix} \quad (2)$$

Considering orthogonal filters for DTCWT computation the length of filters for $\{h_0(n), f_1(0)\}$ and $\{h_1(n), f_0(0)\}$ will be equal resulting in satisfying para-unitary property as in Eq. (3) and time reversal property as in Eq. (4).

$$\sum_n h_i[n]\ h_j[n+2k] = \delta[i-j]\delta[k]$$

$$\sum_n g_i[n]\ g_j[n+2k] = \delta[i-j]\delta[k] \quad (3)$$

$$h_1[n] = (-1)^n h_0[L-n-1]$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$g_1[n] = (-1)^n g_0[L - n - 1]$ (4)

L is the length of the filters and if the filters are biorthogonal then the length of the filters are different. One of the important properties of biorthogonal properties are the time reversal relation as in Eq. (5).

$g_0[n] = h_0[L_0 - n - 1]$
$p_0[n] = f_0[L_1 - n - 1]$ (5)

Considering the Kingsbury's Q-shift filters $\{G_0(z), G_1(z)\}$ are time reversal of $\{H_0(z), H_1(z)\}$. The Polyphase factorization for this filter as presented by Adeel Abbas and Trac D. Tran in 2008 is given as in Eq. (6) for Hp(z) and Eq. (7) for Gp(z) for the 14-tap Q-shift filter proposed by Kingsbury[13].

$$H_P(z) = \begin{bmatrix} -5/64 & 0 \\ 0 & -5/64 \end{bmatrix} \begin{bmatrix} 1 & 3/16 \\ -3/16z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & -37/8 \\ 37/8z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & -5/2 \\ 5/2z^{-1} & z^{-1} \end{bmatrix}$$ (6)

$$G_P(z) = \begin{bmatrix} -1/64 & 0 \\ 0 & -1/64 \end{bmatrix} \begin{bmatrix} 1 & 85/16 \\ -85/16z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & 37/8 \\ -37/8z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & 5/2 \\ -5/2z^{-1} & z^{-1} \end{bmatrix}$$ (7)

Kingsburys 14-tap Q-shift filter is orthogonal and the real part of this filter is time reverse of the imaginary part satisfying $\Psi g(t) = \Psi h(N-1-t)$ and $G_0(z)$ is time-reversal of $H_0(z)$. Detailed discussion on multiplierless approximation for lifting scheme coefficients is presented by Ying-Jui Chen et al. in 2002 [14]. Considering 10-tap Qshift filters for DTCWT polyphase factorization scheme is used to arrive at lifting scheme coefficients.

2. DERIVING LIFTING SCHEME FOR 10-TAP DTCWT FILTER

At every level of DTCWT decomposition row processing and column processing is carried out considering four filters representing real and imaginary filter banks. For 2D image registration, level-1 decomposition using DTCWT generated 16 sub bands of which 12 of them were high pass sub bands and 4 of the were low pass sub bands. For 3D image registration, level-1 decomposition generated 64 sub bands of which 56 of them were high pass sub bands and 8 of them were low pass sub bands. Computation complexity of filter banks implementation on hardware platform is estimated considering arithmetic units and memory units. Figure 1 presents the filter bank for analysis and synthesis filters for DTCWT. The analysis filters are represented as {H0, H1, G0, G1} and the synthesis filters are {H'0, H'1, G'0, G'1}. The corresponding filter coefficients for analysis filters are presented in Table 1.

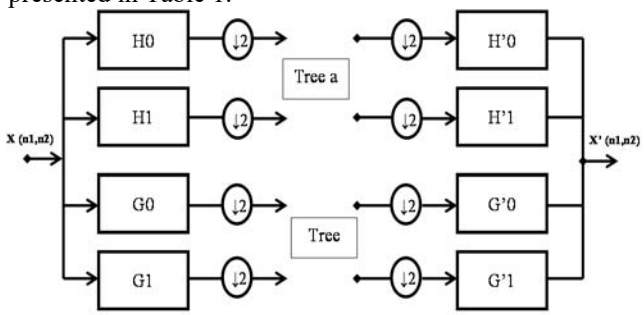

Figure 1: DTCWT analysis and synthesis filters

The 10-tap filter coefficients represent Kingsbury Q-shift filters and it is required to design high speed area efficient and low power architecture for implementation of the filter structure on FPGA platform. The synthesis filters coefficients for Kingsbury Q-shift 10-tap filter is presented in Table 2. Based on the discussions presented in previous section, lifting scheme based algorithm is designed and suitable architecture is developed for FPGA implementation.

Table 1: Analysis filter coefficients

n	H0	H1	G0	G1
1	0.05113040528	-0.0061818818921	-0.006181881892	-0.05113040528383
2	-0.0139753702468	0.00168968127252	-0.00168968127252	-0.01397537024688
3	-0.109836051665	-0.100231219507	-0.100231219507	0.1098360516659
4	0.263839561058	-0.000873622695217	0.000873622695217	0.2638395610589
5	0.766628467793	0.563655710127	0.563655710127	-0.7666284677930
6	0.5636557101270	-0.766628467793	0.766628467793	0.5636557101270
7	0.00087362269521	0.263839561058	0.263839561058	-0.000873622695217
8	-0.100231219507	0.109836051665	-0.109836051665	-0.100231219507
9	-0.0016896812725	-0.0139753702468	-0.0139753702468	0.00168968127252
10	-0.0061818818921	-0.0511304052838	0.0511304052838	-0.0061818818921

Table 2: Synthesis filter coefficients

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

n	H'0	H'1	G'0	
1	-0.0061818818921	-0.051130405283	0.051130405283	-0.0061
2	-0.0016896812725	-0.013975370246	-0.0139753702468	0.0016
3	-0.100231219507	0.109836051665	-0.10983605166	-0.100
4	0.0008736226952	0.263839561058	0.263839561058	-0.0008
5	0.563655710127	-0.766628467793	0.76662846779	0.563
6	0.766628467793	0.563655710127	0.563655710127	-0.76
7	0.263839561058	-0.0008736226952	0.00087362269521	0.263
8	-0.109836051665	-0.100231219507	-0.10023121950	0.109
9	-0.013975370246	0.0016896812725	-0.0016896812725	-0.013
10	0.0511304052838	-0.0061818818921	-0.0061818818921	-0.051

The analysis filter coefficients simplified using approximation algorithm in MATLAB and the corresponding lifting scheme coefficients are derived and is presented in Figure 2. The lifting scheme method derived for the 10-tap filters requires three stages of predict-update pair of arithmetic operations and the final stage requires scaling process to generate the filter coefficients

```
LS = {...
'p'              [  0.27332798]              [0]
'd'              [ -0.25432762  3.95929757]  [0]
'p'              [ -0.08290817 -0.23935050]  [2]
'd'              [  8.25953617-12.26138873]  [-2]
'p'              [ -0.25502421  0.06746501]  [4]
'd'              [  3.86470280]              [-4]
[ -0.68356337]   [ -1.46292215]              []
};
```

Figure 2: Lifting scheme coefficients for 10-tap filter

Predict and update filter coefficients obtained is found to be in the range of |0.06746501| to |12.26138873| with dynamic range variation of 99.44%. Using Laurent polynomial predict and update expressions for lifting scheme is presented in Eq. (8).

P1(z) = 0.2733

P2(z) = - 0.2543 + 3.959*z^(-1)

P3(z) = - 0.08291*z^(+2) - 0.2394*z^(+1)

P4(z) = + 8.26*z^(-2) - 12.26*z^(-3)

P5(z) = - 0.255*z^(+4) + 0.06747*z^(+3)

P6(z) = 3.865*z^(-4)

(8)

K1 = -0.6836 and K2 = -1.4629

The polyphase factorization for the 10-tap analysis filter $H_{p10}(z)$ is presented in Eq. (9), similarly the Polyphase factorization for 10-tap filter $G_{p10}(z)$ is also arrived at.

```
        | 1      0.2733  |
        |                |
  val = |                |
        |                |
        | 0        1     |

|          1             0 |
|                          |
|                          |
|                          |
| - 0.2543 + 3.959*z^(-1)  1 |

        | 1    - 0.08291*z^(+2) - 0.2394*z^(+1) |
        |                                       |
  val = |                                       |
        |                                       |
        | 0                 1                   |

        |              1                  0 |
        |                                   |
  val = |                                   |
        |                                   |
        | + 8.26*z^(-2) - 12.26*z^(-3)    1 |

        | 1    - 0.255*z^(+4) + 0.06747*z^(+3) |
        |                                      |
  val = |                                      |
        |                                      |
        | 0                 1                  |

        |        1              0  |
        |                          |
  val = |                          |
        |                          |
        | 3.865*z^(-4)          1  |
```

(9)

It is required to design efficient architecture for implementation of DTCWT filters considering lifting scheme algorithm. The filter coefficients are fractional numbers and require 16 bit number representation. Representing the filter coefficients using integer number representation simplifies the complexity of arithmetic unit. The lifting scheme coefficients for both real tree and imaginary tree (tree a and tree b) are scaled by a factor of 16 (24) and rounded off to its nearest integer. The scaled and rounded lifting scheme coefficients represented as $\{H_{Q0}, H_{Q1}, G_{Q0}, G_{Q1}\}$ are presented in Table 3 for the real tree filters and Table 4 for imaginary tree filters. The maximum and minimum values of filter coefficients after scaling are 196 and zero. The dynamic range variation is approximately

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

99.999% after scaling and rounding. The dynamic range before and after scaling and rounding is changed by a factor of 0.00559, indicating a very small change in magnitude variations

Table 3 :Quantized filter coefficients for real tree

Tree a				
N	Lifting coefficients		Quantized lifting coefficients	
	H_0	H_1	H_{Q0}	H_{Q1}
P1	4.3732	0	4	0
P2	-4.0692	63.3487	-4	63
P3	-1.3265	-3.8296	-1	-4
P4	132.1525	-196.1822	132	-196
P5	-4.0803	1.0794	-4	1
P6	61.8352	0	62	0
Ka	-10.9370	-23.4067	-11	-23

Table 4: Quantized filter coefficients for imaginary tree

Tree B				
N	Lifting coefficients		Quantized lifting coefficients	
	G_0	G_1	G_{Q0}	G_{Q1}
Q1	-0.2733	0	-4	0
Q2	0.2543	-3.9592	4	-63
Q3	0.0829	0.2393	1	4
Q4	-8.2595	12.2613	-132	196
Q5	-0.0037	-0.0674	0	-1
Q6	3.8647	0	62	0
Kb	5.6537	0.1768	90	3

Figure 3 is the lifting scheme structure for the real tree filter bank of DTCWT. The input sequence x(n) is split into even and odd samples and is processed by the three stages of predict-update arithmetic operation and then is finally processed by the scaling operations. The terms s(z) and t(z) are generically used for representation for predict and update operations respectively. The Polyphase factorization for the structure is expressed in terms of s(z) and t(z) as in Eq.(10).

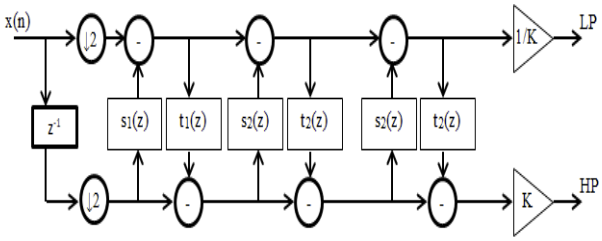

Figure 3: Generic representation of 10-tap lifting scheme structure

$$P(2) = \prod_{i=1}^{n} \begin{bmatrix} 1 & S_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (10)$$

Figure 4 presents the proposed lifting scheme structure derived from the generic structure in Figure 3. The predict and update terms are represented as P(z) and is represented in terms of Polyphase factorization as in Eq. (11). The proposed lifting scheme structure consists of two subtractors and two multipliers every stage of predict-update operations. The last stage is multiplication operations. Figure 5 is the lifting scheme structure for tree b.

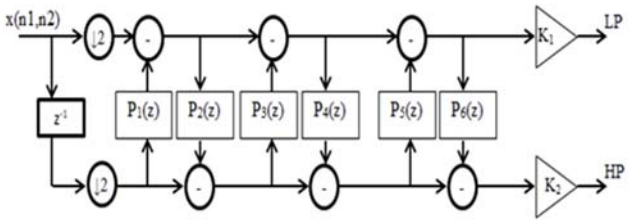

Figure 4: Lifting scheme structure for real tree or tree a

$$P_a(z) = \begin{bmatrix} 1 & P_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & P_3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P_4 & 1 \end{bmatrix} \begin{bmatrix} 1 & P_5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P_6 & 1 \end{bmatrix} \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \quad (11)$$

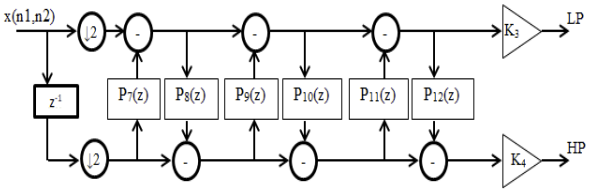

Figure. 5: Lifting scheme structure for real tree or tree b

The number of arithmetic operations required for implementing DTCWT based on lifting scheme is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

further optimized to reduce power and area requirement and improve processing speed.

4. DESIGN OF EFFICIENT LIFTING SCHEME STRUCTURE

Based on the polyphase factorization expressions the step by step operations of lifting scheme algorithm is presented in Table 5 for tree a and Table 6 for tree b. There are eight steps starting from split operation to scaling operation. The split operation is performed by down sampling by 2 and using one clock cycle delay. The even term is represented by X_{2i} and X_{2i+1} represent the odd term and are represented as s_i^0 and d_i^0 respectively. The lifting filter coefficients for tree a are represented as {a1, b1, b2, d1, d2, e1, e2, f1} and for tree b is represented as {a2, b3, b4, d3, d4, e3, e4, f2}.

Table 5: Step by step process of lifting scheme algorithm for tree a

Lifting steps	Arithmetic Expressions	With lifting coefficients
Even and odd split	$S_i^0 = X_{2i}$ & $d_i^0 = X_{2i+1}$	$S_i^0 = X_{2i}$ $d_i^0 = X_{2i+1}$
Predict 1	$d_i^1 = d_i^0 + a_1 S_i^0$	$d_i^1 = d_i^0 + 4 S_i^0$
Update 1	$S_i^1 = S_i^0 + b_1 d_i^1 + b_2 d^1_{i+1}$	$S_i^1 = S_i^0 - 4 d_i^1 + 63 d^1_{i+1}$
Predict 2	$d_i^2 = d_i^1 + c_1 S^1_{i-2} + c_2 S^1_{i-1}$	$d_i^2 = d_i^1 - S^1_{i-2} - 4S^1_{i-1}$
Update 2	$S_i^2 = S_i^1 + d_1 d^2_{i+2} + d_2 d^2_{i+3}$	$S_i^2 = S_i^1 + 132\, d^2_{i+2} - 196\, d^2_{i+3}$
Predict 3	$d_i^3 = d_i^2 + e_1 S^2_{i-4} + e_2 S^2_{i-3}$	$d_i^3 = d_i^2 - S^2_{i-4} + S^2_{i-3}$
Update 4	$S_i^3 = S_i^2 + f_1 d^3_{i+4}$	$S_i^3 = S_i^2 + 62\, d^3_{i+4}$
Scaling	$S_i = k_1 S_i^3$ $d_i = k_2 d_i^3$	$S_i = -11\, S_i^3$ $d_i = -23\, d_i^3$

Table 6: Step by step process of lifting scheme algorithm for tree b

Lifting steps	Arithmetic Expressions	With lifting coefficients
Even and odd split	$S_i^0 = X_{2i}$ & $d_i^0 = X_{2i+1}$	$S_i^0 = X_{2i}$ $d_i^0 = X_{2i+1}$
Predict 1	$d_i^1 = d_i^0 + a_2 S_i^0$	$d_i^1 = d_i^0 - 4 S_i^0$
Update 1	$S_i^1 = S_i^0 + b_3 d_i^1 + b_4 d^1_{i+1}$	$S_i^1 = S_i^0 + 4 d_i^1 - 63 d^1_{i+1}$
Predict 2	$d_i^2 = d_i^1 + c_3 S^1_{i-2} + c_4 S^1_{i-1}$	$d_i^2 = d_i^1 + S^1_{i-2} + 4S^1_{i-1}$
Update 2	$S_i^2 = S_i^1 + d_3 d^2_{i+2} + d_4 d^2_{i+3}$	$S_i^2 = S_i^1 - 132\, d^2_{i+2} + 196\, d^2_{i+3}$
Predict 3	$d_i^3 = d_i^2 + e_3 S^2_{i-4} + e_4 S^2_{i-3}$	$d_i^3 = d_i^2 - S^2_{i-4} - S^2_{i-3}$
Update 4	$S_i^3 = S_i^2 + f_2 d^3_{i+4}$	$S_i^3 = S_i^2 + 62\, d^3_{i+4}$
Scaling	$S_i = k_3 S_i^3$ $d_i = k_4 d_i^3$	$S_i = 90\, S_i^3$ $d_i = 3\, d_i^3$

In step 1 the input samples are split into even and odd samples and are denoted as Si0 and di0. In step 2 of the lifting scheme the even and odd samples are processed to predict the term di1, in this expression the lifting constant (a1) derived in the previous section and presented in Table 5.7 is used. The constant a1 is dyadic rational number and performing multiplication with the multiplicand si0 in step 1 is carried out by left shifting the multiplicand by 2 bits. The multiplication operation in step 2 is converted to a multiplierless operation. In step 3, update 1 operation is carried out and requires two samples of predict 1 phase (d0i and d1i+1) and two lifting constants b1 and b2. Computation of si1 requires two multiplication and two addition operations. Considering two samples for i=0 and i=1, the update 1 expression is presented in Eq. (12).

$$S_0^1 = S_0^0 - 4\, d_0^1 + 63\, d^1_1 \quad (i=0) \quad (12a)$$

$$S_1^1 = S_1^0 - 4\, d_1^1 + 63\, d^1_2 \quad (i=1) \quad (12b)$$

The term d11 is used twice once for computing S01 and s11. In computing s10, d11 is multiplied by 63 and in computing s11 it is multiplied by 4. Since multiplication is carried out twice on the same term a modified logic is presented in this work so as to reuse the multiplied partial products. Table 7 presents the modified expression of lifting scheme

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(middle column) and the split lifting expression (last column) for performing reuse operations.

Table 7: Modified and split lifting schemes

Steps	With lifting coefficients	Modified lifting expression	Split lifting expressions
1	$S_i^0 = X_{2i}$ $d_i^0 = X_{2i+1}$	$S_i^0 = X_{2i}$ $d_i^0 = X_{2i+1}$	-
2	$d_i^1 = d_i^0 + 4\ S_i^0$	$d_i^1 = d_i^0 + 4\ S_i^0$	-
3	$S_i^1 = S_i^0 - 4\ d_i^1 + 63\ d^1_{i+1}$	$S_i^1 = S_i^0 - 4\ d_i^1 + 64\ d^1_{i+1}$	$S_i^1 = S_i^0 - 4\ d_i^1 + 4\ d^1_{i+1} + 60\ d^1_{i+1}$
4	$d_i^2 = d_i^1 - S^1_{i-2} - 4S^1_{i-1}$	$d_i^2 = d_i^1 - S^1_{i-2} - 4S^1_{i-1}$	-
5	$S_i^2 = S_i^1 + 132\ d^2_{i+2} - 196\ d^2_{i+3}$	$S_i^2 = S_i^1 + 128\ d^2_{i+2} - 192\ d^2_{i+3}$	$S_i^2 = S_i^1 + 64\ d^2_{i+2} + 64\ d^2_{i+2} - 64\ d^2_{i+3} - 128\ d^2_{i+3}$
6	$d_i^3 = d_i^2 - 4\ S^2_{i-4} + S^2_{i-3}$	$d_i^3 = d_i^2 - 4\ S^2_{i-4} + S^2_{i-3}$	-
7	$S_i^3 = S_i^2 + 62\ d^3_{i+4}$	$S_i^3 = S_i^2 + 56\ d^3_{i+4}$	$S_i^3 = S_i^2 + 16\ d^3_{i+4} + 8\ d^3_{i+4} + 32\ d^3_{i+4}$
8	$S_i = -11\ S_i^3$ $d_i = -23\ d_i^3$	$S_i = -10\ S_i^3$ $d_i = -24\ d_i^3$	$S_i = -10\ S_i^3$ $d_i = -16\ d_i^3 - 8\ d_i^3$

The lifting coefficients in step 3, step 5, step 7 and step 8 are replaced with constants that are dyadic rational numbers. The split expressions for modified lifting scheme are rewritten considering the constants in multiples of dyadic rational numbers. Considering the modified and split lifting logic presented in Table 7 the lifting expression in Eq. (12) is rewritten as in Eq. (13) considering two time intervals of i=0 and i=1.

$$S_0^1 = S_0^0 - 4\ d_0^1 + 64\ d^1_1 \quad (i=0) \qquad (13a)$$

$$S_1^1 = S_1^0 - 4\ d_1^1 + 64\ d^1_2 \quad (i=1) \qquad (13b)$$

The data flow diagram or graph for the expression in Eq. (13) is presented in Figure 6. Rounding of the coefficient to 64 and splitting the constant into two numbers as 4 and 60 the data flow graph is redrawn. The advantage of this logic is the lifting term d11 multiplied by the constant 4 is used twice to compute s01 and s11 saving one multiplication operation.

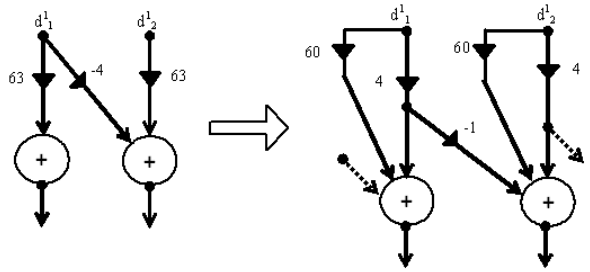

Figure 6: Dataflow graph for step 3 using split lifting expression

Multiplying the input data by dyadic twiddle factor of 4 is performed by shifting left the multiplicand by 2-bits. Multiplication by the constant 60 is carried out by performing left shift operation of the multiplicand. Since 60 is not a dyadic integer, the constant 60 is expressed as (64-4). The binary representation of 60 is "011 1100", this implies there are 4 ones and hence requires four left shift operations. Representing the constant 60 by 64 ("1000 0000") and 4 ("0000 0100") the number of ones are reduced to two. The arithmetic operations in of shifting left the multiplicand when multiplied by 64 and 4 will be appending 6 zeros and 2 zeros at the LSB respectively. After appending operation, the partial products are added using one adder. The total number of shifting operations and addition operations are reduced in this method of data processing.

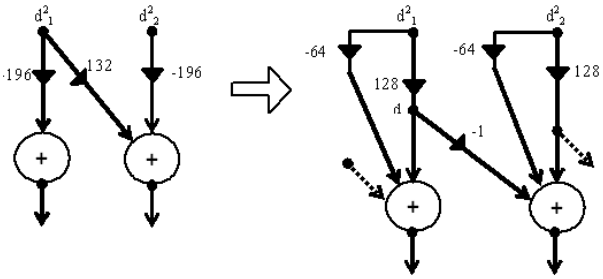

Figure 7: Dataflow graph for step 5 using split lifting expression

Figure 7 presents the data flow diagram for step 5 of split lifting scheme. The lifting coefficient -196 is rounded off to nearest integer -192 and is split into -64 and 128. Similarly the lifting coefficient 132 is rounded off to 128. Multiplying the term d21 with 128 in the first half of data flow graph is carried out by appending 7 zeros on the LSB and the product is represented as 'd'. The product term

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

'd' is used in two branches of the data flow diagram and is further multiplied by -64 (carried out by appending 7 zeros on LSB) and 1. By realizing the data flow graph by split logic the number of bits required for representing the data is limited to 8 bits (including sign bit) and the product term generated in the first half of the data flow graph (represented by 'd') is reused in the right flowing data graph thus saving one arithmetic operation.

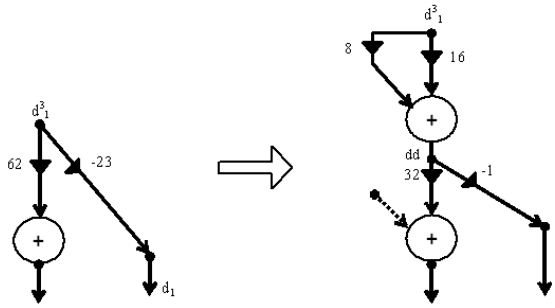

Figure 8: Dataflow graph for step 8 using split lifting expression

Figure 8 presents the data flow graph for step 8 of proposed lifting scheme method. In this step the lifting coefficient 62 is rounded off to 56 and the coefficient -23 is rounded of to -24. The constant 56 is split into three constants of 16, 8 and 32. Multiplying the term d31 by 16 and 8 in the first half of the data flow graph is carried out by appending the input data d31 first by 4 zeros on the LSB and then by appending by 3 zeros to generate the intermediate term represented by 'dd'. The term 'dd' is used by the second half of the data flow graph to perform multiplication by 32 (carried out by appending by 5 zeros on LSB) and sign reversal operation in the right flowing data flow graph. The proposed lifting scheme method for the 10-tap filter represented in Table 7 designed for tree a is presented in Figure 9. Figure 10 presents the proposed lifting scheme method for tree b.

In the proposed lifting scheme structure the concept of reusability is adopted to reuse the partial products for computing lifting scheme predict and update intermediate outputs. The number of adders for realising one pair of low pass and high pass filter outputs are computed to be of 12 for both tree a and tree b. Computing both real tree and imaginary tree outputs the lifting scheme structure are similar only the sign of the lifting scheme coefficients change at each stage of predict and update steps.

5. FUNCTIONAL VERIFICATION OF SPLIT LIFTING STRUCTURE

Split lifting structure developed in the previous chapter is verified for its logic correctness by considering test vectors with range 0 to 255. In order to avoid overflow, the input vector is considered in the range of 0 to 127. A know vector set of 10 pixels is considered and the data is processed by the 10-tap filter using convolution operation. Output generated in this process using MATLAB environment is considered as reference output data set. It is required to obtain the same set of outputs after the known set of inputs are processed by the lifting scheme structure. The lifting logic is modelled using Verilog using hierarchical approach, the arithmetic units are modelled using behavioural model and the arithmetic models are integrated into top model using structural modelling. Test bench is developed to verify the model and the test bench is loaded with the known test vectors. The Verilog model is simulated in Xilinx ISE and the results are obtained in the simulator window. The output generated in binary format is converted to signed integer format and the outputs are noted down to compare with the results of outputs obtained in the MATLAB model. For a known set of test inputs theoretical values of the actual output is calculated and is compared with the simulation results output. The split lifting scheme logic is modelled as 1D-DTCWT processor and the results for known set of inputs in the range of 0 to 127 have been used as test inputs, and the corresponding outputs are obtained using ISE Sim. The test results are shown in Figure 9. From the simulation results it is found that the 1D-DTCWT processor results are matching the theoretical requirements.

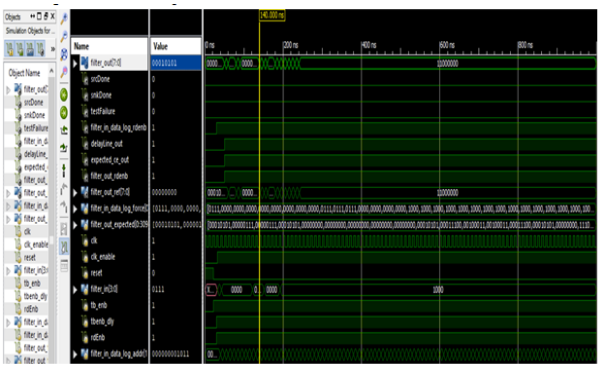

Figure 12: First stage simulation results of 1D DWT processor using Model Sim

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The input pixels are sent into the DTCWT engine as given in Table 8 in "DATA IN" vector. The tree a and tree b filter coefficients used for the filter design are indicated along with the MATLAB results and HDL results.

Table 8: Simulation results for known test case of input vector

	1D-DTCWT tree a			1D-DTCWT tree b		
In	H_{Q0}	Output (MATLAB)	Out put (HDL)	H_{Q1}	Output (MATLAB)	Out put (HDL)
42	4	168	164	0	0	0
42	-4	-168	-164	63	2646	2648
42	-1	-42	-42	-4	-168	-164
42	132	5544	5548	-196	-8232	-8234
42	-4	-168	-164	1	42	42
42	62	2604	2602	0	0	0
42	-11	-462	-464	-23	-966	-968

The input and output waveform for the simulation of DTCWT engine is evaluated the simulation result matches the computation results in Table 8. Software model results and hardware model results are compared for different test vectors and the results are found to be similar demonstrating the functionality of architecture designed. The DTCWT engine takes 30 clock periods to compute the tree a and tree b output coefficients at the starting of each row. The latency is estimated between the low \to high transition of sync signal to first output data valued signal. The DTCWT takes only 22 clock cycles to computation the DTCWT of each line after the computation of first DWT coefficient at the starting each new row.

5.1 FPGA Implementation

The Verilog model is successfully verified for its functionality for both as 1D-DTCWT processor and for 3D-DTCWT processor. The functionally verified model is synthesized using Xilinx ISE targeting Spartan 6 FPGA device. The synthesis results for the 1D-DTCWT processor is shown in Figure 13. The top level model for the 3D-DTCWT processor is shown in Figure 14a and the corresponding internal block diagram after level-1 hierarchy is presented in Figure 14b.

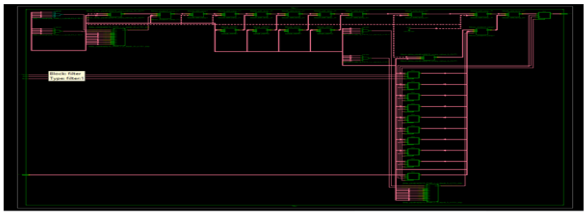

Figure 13: Synthesized netlist for 1D-DTCWT processor

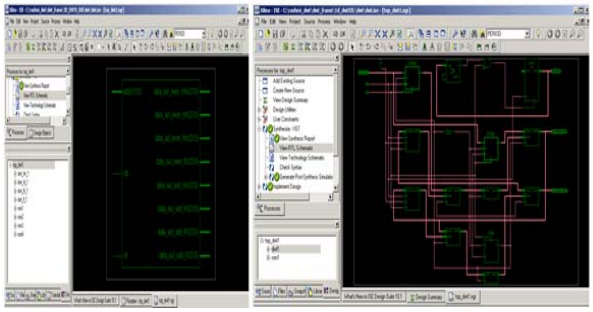

Figure 14a and 14b: Synthesized netlist for 3D-DTCWT model

Table 9 compares the FPGA performance results of the proposed structure with all other implementation structure. From the comparison report, the proposed split lifting scheme based 1D-DTCWT processor is found to operate ate maximum frequency of 342.38 MHz consuming power of less than 28 W. The most efficient implementation scheme is the Distributed Arithmetic (DA) method [15] and the proposed method is demonstrating performances better than the DA method. 23% of improvement in operating frequency and 46% reduction in power dissipation is achieved for the proposed architecture as compared with DA method. Table 10 compares the performances of 3D-DTCWT implementation over Spartan-6 device considering three reference designs. Power dissipation in all three reference architectures is limited to less than 10 W and the results obtained are for single stage DTCWT processing based on proposed split lifting scheme is greater than 56W as the proposed architecture processes nine frames simultaneously. The algorithm proposed in [16] is modelled and is extended for DTCWT computation of 256 x 256 image and the results are compared with proposed split lifting based architecture for 3D DTCWT computation. The hybrid DA architecture discussed in [17] optimizes area utilization on CLBs and DTCWT structure is implemented on Spartan-6 FPGA for four filters. It is observed that the parallel processing algorithm that is designed in this work is able to process data at higher frequency of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

operation as compared with all other architectures. The power dissipation is found to be higher than the existing designs. Development of image registration algorithm on FPGA platform has demonstrated that the total delay in performing 3D-DTCWT is less than 300ps.

6. CONCLUSION

For high speed image processing requirement such as in image registration process that is required in portable devices or hand held devices the operating frequency of over 300 MHz provides 3D image processing at more than 200 frames per second which is demonstrated in this work based on systolic array architecture and DA algorithm. DTCWT based image processing algorithm demonstrating advantages over DWT based image processing this work discusses hardware accelerators for meeting the processing speed and optimum area for hardware implementation. The DTCWT processing module presented in this work is suitable for image decomposition into complex bands providing information on six directional orientations. The proposed architecture can be used as an IP module for image processing applications to be developed on hardware platforms.

REFERENCES:

[1] Selesnick, R. G. Baraniuk, and N. C. Kingsbury, "The dualtree complex wavelet transform," IEEE Signal Processing Magazine, vol. 22, no. 6, 2005 pp. 123–151

[2] Selesnick and K. Y. Li, "Video denoising using 2D and 3D dual-tree complex wavelet transforms," in Wavelets: Applications in Signal and Image Processing X, vol. 5207 of Proceedings of SPIE, San Diego, Calif, USA, August 2003,.pp. 607–618

[3] Jeremy Fowers, Greg Brown, Patrick Cooke, and Greg Stitt. 2012. A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. In Proceedings of the ACM/SIGDA 20th International Symposium on Field Programmable Gate Arrays (FPGA'12). ACM, New York, NY, 47–56.

[4] Deepayan Bhowmik, Paulo Garcia, Andrew M. Wallace, Robert J. Stewart, and Greg Michaelson. 2017. Power efficient dataflow design for a heterogeneous smart camera architecture. In Proceedings of the 2017 Conference on Design and Architectures for Signal and Image Processing (DASIP'17). IEEE, Los Alamitos, CA, 1–6

[5] Robert Stewart, Greg J. Michaelson, Deepayan Bhowmik, Paulo Garcia, and AndyWallace. 2016. A dataflow IR for memory efficient RIPL compilation to FPGAs. In Algorithms and Architectures for Parallel Processing. Lecture Notes in Computer Science, Vol. 1194. Springer, 174–188

[6] S. Ahmad, V. Boppana, I. Ganusov, V. Kathail, V. Rajagopalan, and R. Wittig. 2016. A 16-nm multiprocessing system-on-chip field-programmable gate array platform. IEEE Micro 36, 2, 48–62

[7] J. Jeddeloh and B. Keeth. 2012. Hybrid Memory Cube new DRAM architecture increases density and performance. In Proceedings of the 2012 Symposium on VLSI Technology (VLSIT'12). IEEE, Los Alamitos, CA, 87–88

[8] Erik Jan Marinissen and Yervant Zorian. 2017. Guest editors introduction: Design and test of a high-volume 3-D stacked graphics processor with high-bandwidth memory. IEEE Design and Test 34, 1, 6–7

[9] Ioannis Stratakos, Dimitrios Gourounas, Vasileios Tsoutsouras, Theodore Economopoulos, George Matsopoulos and Dimitrios Soudris, Hardware Acceleration of Image Registration Algorithm on FPGA-based Systems on Chip COINS, May 5–7, 2019, Crete, Greece

[10] ROBERT STEWART, KIRSTY DUNCAN, GREG MICHAELSON, and PAULO GARCIA, DEEPAYAN BHOWMIK and ANDREW WALLACE, RIPL: A Parallel Image Processing Language for FPGAs, ACM Transactions on Reconfigurable Technology and Systems, Vol. 11, No. 1, Article 7, March 2018

[11] Sweldens, W.: The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Construction. Proc. Of SPIE, Vol.2569, San Diego, USA, July 1995, 68–79

[12] Daubechies, I.; Sweldens, W.: Factoring Wavelet Transform into Lifting Steps. J. Fourier Anal. Appl., Vol.4, No.3, 1998, 247–269

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[13] N. G. Kingsbury, "Design of Q-shift complex wavelets for image processing using frequency domain energy minimisation," Proc. IEEE ICIP, Sep. 2003,pp. 1013–1016,

[14] Y. J. Chen, S. Oraintara, T. D. Tran, K. Amaratunga, and T. Q. Nguyen, "Multiplierless approximation of transforms with adder constraint," IEEE Signal Process. Lett., vol. 9, Nov. 2002, pp. 344–347,

[15] M. M. Eshtawie and M. Othman, "Distributed Arithmetic Implementation of an Optimized Raised Cosine FIR Filter Coefficients," 2006 8th international Conference on Signal Processing, Beijing, 2006, doi: 10.1109/ICOSP.2006.345498.

[16] Memory Efficient High Speed Systolic Array Architecture Design with Multiplexed Distributed Arithmetic for 2D DTCWT Computation on FPGA, Poornima B., Sumathi A., Cyril Prasanna Raj P

[17] S. S. Divakara, Sudarshan Patilkulkarni, Cyril Prasanna Raj, "High Speed Area Optimized Hybrid DA Architecture for 2D-DTCWT" International Journal of Image and Graphics Vol. 18, No. 1, 2018

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 9: Comparison of FPGA metrics for 1D-DTCWT processing

Parameters	Fully-Parallel	Fully-Serial	Partly-Serial	Cascade-Serial	Distributed-Arithmetic	Split Lifting Scheme
Area (Total cell area)	1.37e+09	2.96e+09	4.64e+09	4.62e+06	4.64e+09	0.67e+09
Time(Max Delay)	472.527ps	9185.81ps	8665.33ps	8737.84ps	3817.75ps	292.001ps
Power	118.2441031 W	85.1820803 W	133.1774176W	133.9282344W	52.0805906 W	28.0098 W
Computed Frequency	99.0533M Hz	108.864M Hz	115.402M Hz	114.445M Hz	261.934M Hz	342.38M Hz

Table 10: Comparison of hardware requirements

	SAA with Multiplexed DA [16]	DTCWT [17]	Direct DTCWT	Proposed
Number of Slice Registers	3672	4112	7482	10234
Number of Slice LUTs	3173	4091	7224	10054
Total power (W)	1.5702	1.71111	2.00207	56W
Max. Frequency (MHz)	321.89	289.12	212.67	332.98

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

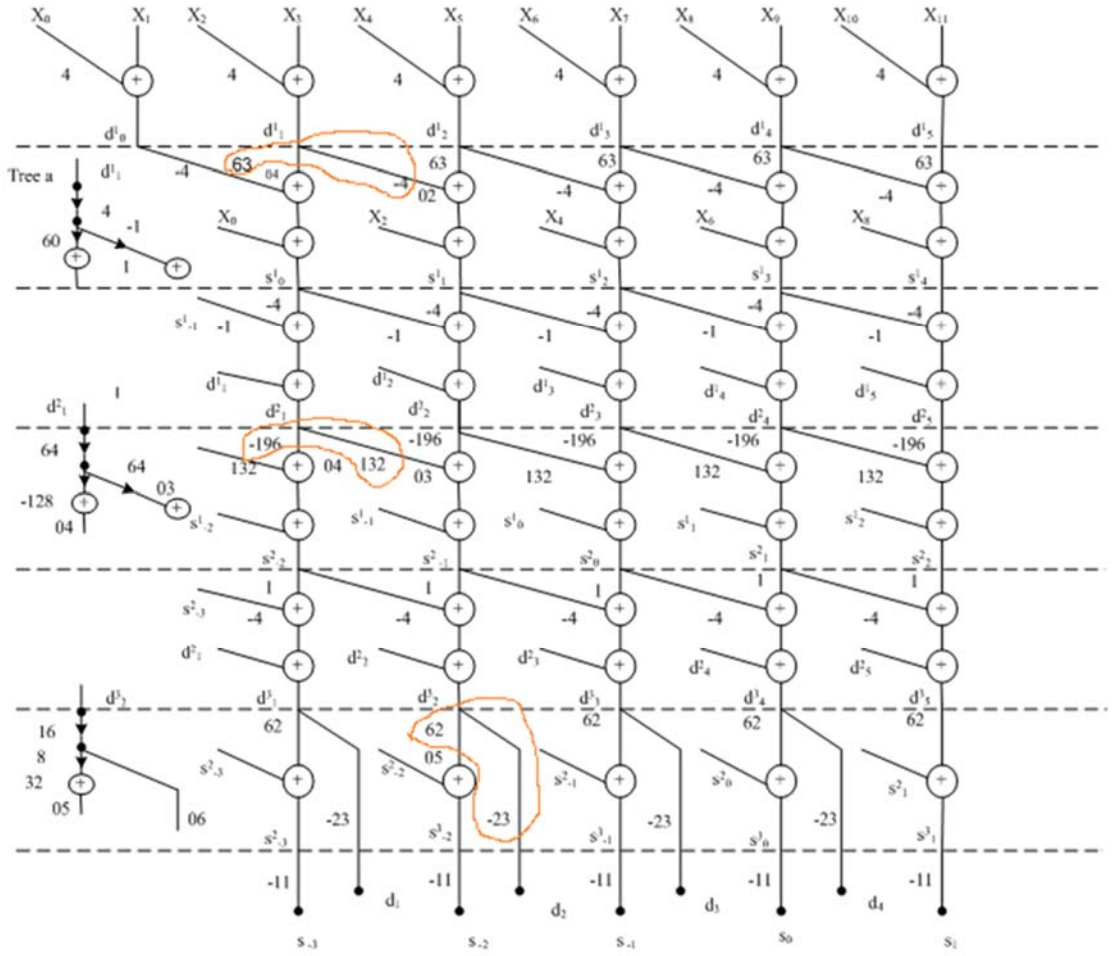

Figure 9: Proposed lifting scheme data flow graph for tree a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

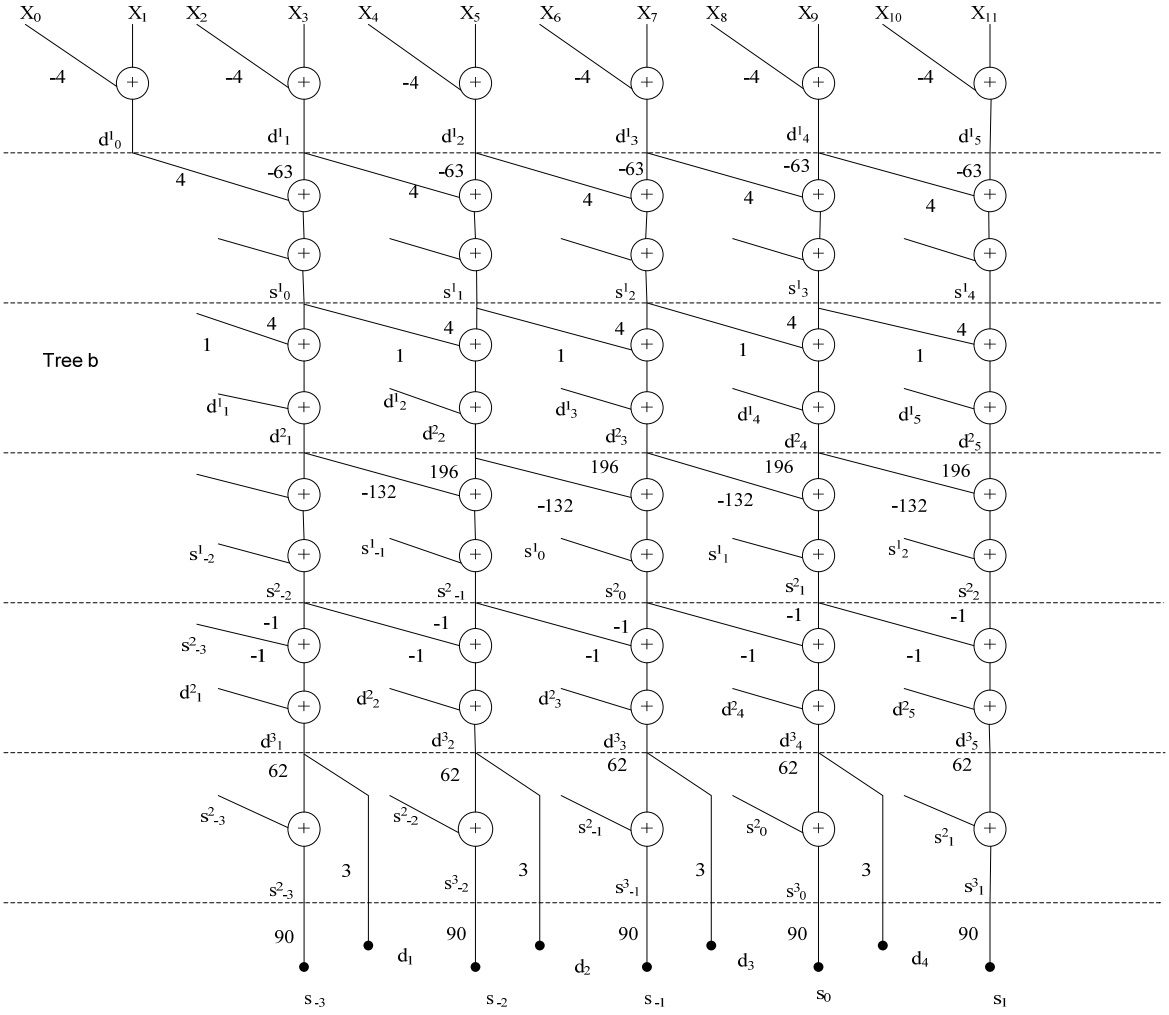

Figure 10: Proposed lifting scheme data flow graph for tree b

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

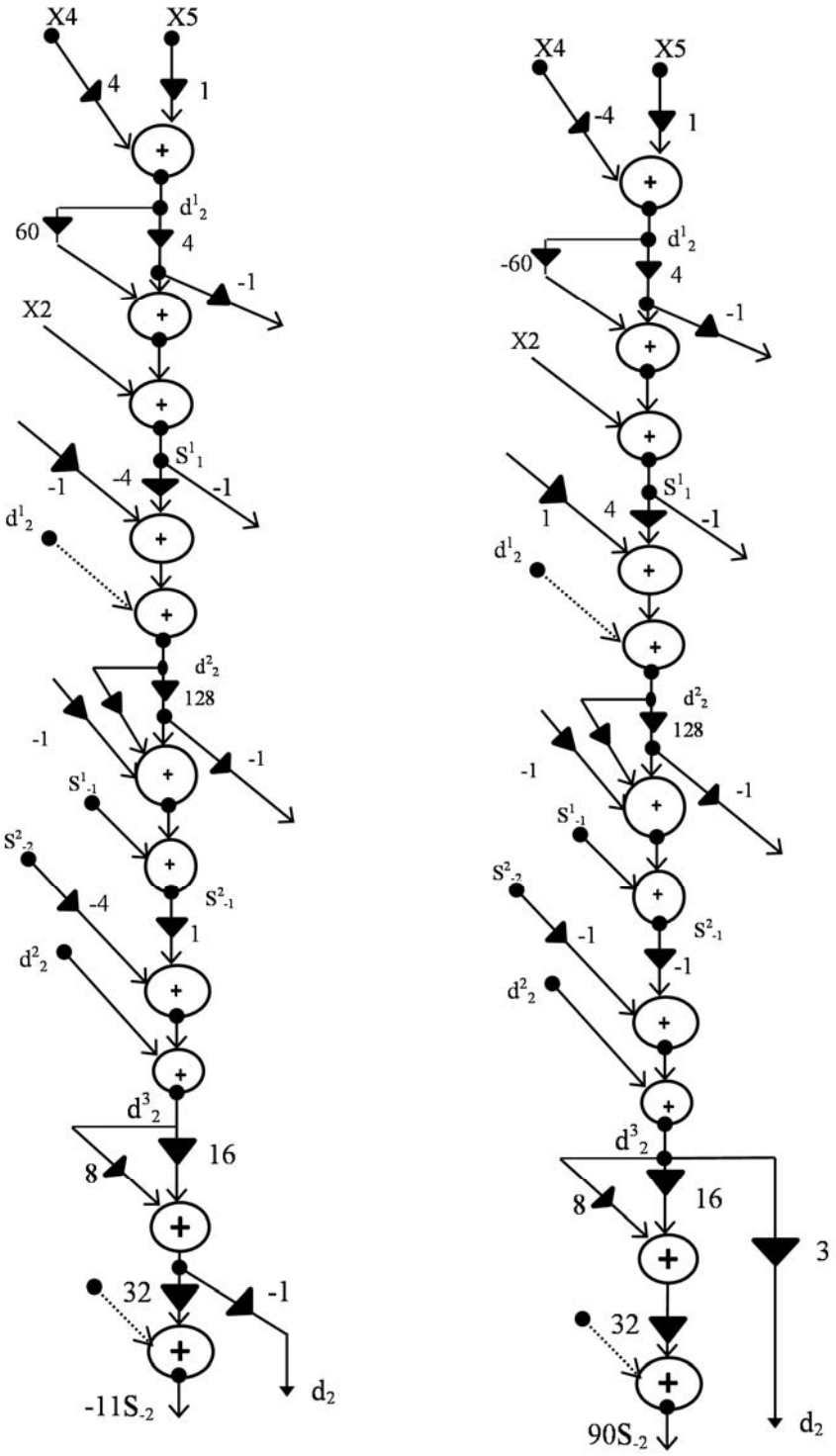

Figure 11: Data flow diagram for computing {S_{-2}, d_2} of tree a and tree b