Received: January 16, 2019

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Power and Area Efficient Radix-8 Booth Multiplier for 2-D DWT Architecture

Gundugonti Kishore Kumar[1]* **Narayanam Balaji[2]** **Kotha Srinivasa Reddy[1]** **Vemu Thanuja[1]**

[1]*Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, India.*
[2]*Jawaharlal Nehru Technological University Kakinada, Narasaraopet, India.*
* Corresponding author's Email: gkishorekumar@vrsiddhartha.ac.in

Abstract: This paper presents a modified 2D Discrete Wavelet Transform (DWT) architecture with a proposed 16-bit Radix8 Booth multiplier. Existing architecture makes use of Canonic Sign Digit (CSD) representation and when replaced the CSD multiplier with the proposed 16-bit Radix8 Booth multiplier it achieves better performance with small area and low power. In proposed Radix-8 Booth multiplier, the necessary product terms are generated and the remaining terms are truncated. In this method, the *n* order bit required by the specific coefficient is obtained and the remaining *n* bits are truncated so that 2*n* bit output truncated to *n* bit. The modified 2D DWT architecture is proposed to enhance that it occupies less number of clock cycles, so that it improves in the speed of operation By comparing synthesis results for existing CSD multiplier and the proposed Radix-8 Booth multiplier achieves an improvement of nearly 29.02% Area Delay Product (ADP) and 26.13% Power Delay Product (PDP).

Keywords: Discrete wavelet transform, Canonic sign digit, Radix-8 booth multiplier, Area delay product, Power delay product.

1. Introduction

Wavelets convert the picture into a progression of wavelets that can be store more effectively than pixel squares, so DWT architectures has gained its importance in applications where scalability and tolerable degradation is main in wavelet coding schemes. When time and frequency domains are averaged for whole duration of the signal gives information about the DWT. 2D DWT is used extensively in many fields of engineering and medical applications such as in biometrics, image analysis and imaging applications such as JPEG 2000 etc.

Several lifting architectures are implemented for effective implementation of 2D DWT. Novel architecture based on flipping structures [1] to reduce critical path by reducing the pipelining stages through rearranging the intermediate values and an effective dual-scan flipping structure is done by using modified data flow graph in serial operation and uses a N2/2 clock cycles in a Z-scan model optimizing the parallel computations with pipeline operation. Some of the architectures based on parallel lifting scheme [2] with effective memory accessing scheme based on scanning method requires less memory. Architecture based on Z-scanning technique [3] presents a multiplier less pipeline architecture to reduce latency and uses a 4N temporal memory. With improvements done to convolution based architectures it provides small overhead of complexity and with no use of temporary registers for storing different values with low area and power reconfigurable architecture [4] using 9/3 and 5/3 filters. For implementing 2D DWT in multi-level with regular structure to maximize hardware utilization efficiency [5] for high throughput and low latency is implemented for efficient memory based implementation. Based on short critical path a lifting based DWT [6] is implemented for efficient memory usage in scanning method. Another convolution based implementation for memory efficient generic structure [7] which computes a three-level 2D DWT based on Daubechies is implemented for high throughput rate is implemented. Implementation of several architectures has their research on memory efficient, with high throughput rate for different types of techniques in which the need for low area and low power [8] makes a demand for various applications. In DWT architectures, design of the efficient multiplier plays a crucial role for performing the operation with approximated and faster output is discussed in Booth multiplier [9,10]. Existing multiplier (CSD) makes use of shift and adds operation for fast computation but it consumes more area and power due to which the need for hardware requirements becomes more. The above mentioned drawbacks are improved in this paper by designing a 16-bit fixed-width Booth multiplier which occupies less area and low power. The reduction in area and power are achieved in this paper by truncating the partial product terms and necessary changes are made in 2D DWT architecture for effective implementation.

This paper is organized as follows. A detailed description on previous related work is presented in Section 2. Implementation of proposed Radix-8 Booth multiplier is explained and by taking an example it is examined in Section 3. Improvements done to 2D DWT architecture is shown in Section 4. The synthesis results of existing and proposed architectures are listed in Section 5 and conclusions made from the work are detailed in Section 6.

2. Related work

C. Wu, W. Zhang and J. Liu [11] proposed a 2D DWT architecture with multi-level without off-chip RAM using the CSD representation for the multiplier coefficients. The CSD makes use of shift and add operation for the implementation of the multiplier. In this 2D DWT structure the architecture is implemented in three stages. First the pixel inputs are stored in RAM input and feed to first stage 2D DWT structure which gives four sub-bands, out of which low-low band occupies large area so they are fed back to next levels which is managed through memory management unit and sort the four-bands. It accomplishes one full adder delay as critical path. In this paper it uses simple shift-add operations but this implementation occupies more area and power and the need for temporal RAM also arises.

Y. K. Lai, L. F. Chen, Y.C. Shih [12] implemented a memory efficient with high performance architecture for 2D DWT lifting based architecture using parallel scanning method. For first level 2D DWT, 4N temporal memory is used to store input data and to store coefficients 9/7 filter is used. This architecture is flexible and the 2D DWT architecture is made of two 1D-DWT with core input and two output coefficients and achieves one multiplier delay as the critical path. By implementing parallel scanning method the internal buffer size and cost of the hardware is small. The problem arises in the latency delay and transposing buffer size is not improved due which number of stages gets increased.

B. K. Mohanty, A. Mahajan and P.K. Meher [13] gave a data access method without utilizing data transposition registers for the calculation of lifting 2D DWT architecture which requires (4N +8P) words for on-chip memory. The architecture is modular and regular used for varying block sizes. It gives a linear array directly from the data dependence graph for parallel and pipeline implementation of 1D DWT. This architecture needs the same calculation resources as in high throughput structures with 1.5N less memory. This structure is proposed for block size of 4 and the drawback is that it occupies more on-chip memory and has more critical path.

A. Darji, S. Agrawal and A. Oza [14] proposed a flipping method with basic control path for implementing efficient modular architecture. The sequential task of lifting data flow is streamlined using parallel calculations with pipeline operation without influencing the critical path. The structure is folded to six multipliers and eight adders to reduce data path. It is symmetrical high speed architecture with low hardware complexity where it has low memory storage requirement. It yields a throughput of two outputs for each cycle with a critical path of one multiplier delay. In this paper it requires more registers for transposing unit and impacts in hardware complexity.

W. Zhang, Z. Jiang, Z. Gao and Y. Liu [15] proposed an architecture for efficient implementation of lifting- based DWT with small area and fast speed. For one multiplier delay a reduction of four pipelining stages are required. In this structure initially the pre-processing unit takes care of converting serial-parallel data and then sent to the column filter for generating four sub-groups which is then fed to transposing unit to satisfy the dataflow order needed by row filter and then scaling is done. This structure has one multiplier delay as critical path having 4N temporal memory size. The requirement of buffer size can be reduced due to which it impacts the memory size.

B. K. Mohanty and A. Choubey [16] implemented a design for 12-bit radix8 Booth multiplier by evacuating an additional row with a little overhead complexity in the conventional radix8 booth multiplier. An adder unit is intended for optimizing the upper most 12 bits for taking the output. The lower-most 12 bits are truncated from the 24 bits with less truncation error. When this multiplier is implemented for the lifting 2D DWT architecture with block based structure it offers less area and power than radix-4 multiplier. In this multiplier design the product terms are generated due to which some more increase in area is obtained which affects the performance.

Many of the 2D DWT architectures listed in literature work are improved in terms of area, power and critical path are discussed which could be improved. To improve the architecture design, a low area and power efficient architecture with an efficient multiplier design is implemented which could result in less area for image processing applications.

3. Implementation of proposed Radix-8 Booth multiplier

Let us consider two k-bit signed numbers A and B, where A is the multiplicand and B is the multiplier which is to be multiplied using Booth algorithm. A and B can be shown in 2's complement as:

$$A = -a_{k-1}2^{k-1} + \sum_{i=0}^{k-2} a_i 2^i \quad (1)$$

$$B = -b_{k-1}2^{k-1} + \sum_{i=0}^{k-2} b_i 2^i \quad (2)$$

Wherea_{k-1}, b_{k-1} of A and B are sign extension bits The booth algorithm distils the k-bit multiplier into *n= [k/3]* groups of three bits. Starting from the right, three bits have to be grouped and for 4th-bit an overlapping bit is considered, where i th digit d_i is defined

$$d_i = \{ b_{3i+2}b_{3i+1}b_{3i}b_{3i-1} \} \quad (3)$$

Where b_{3i-1} is the bit grouped for overlapping that belongs to the (i-1)th digit d_{i-1}. The estimation of d_i is acquired by the equation:

$$d_i = -4b_{3i+2} + 2b_{3i+1} + b_{3i} + b_{3i-1} \quad (4)$$

From the number set {-4, -3, -2, -1, 0, 1, 2, 3, 4} d_i selects the decimal value and when A is multiplied by d_i it forms a product term *(A.d_i)* which results in obtaining necessary partial products of radix8 Booth multiplier are {-4A, -3A, -2A, -A, 0, A, 2A, 3A, 4A}. The acquired partial product term *{2A}* is obtained by shifting A to the left by one bit, *{4A}* is done by left shifting A by two bits and *{3A}* is achieved by summing A and 2A. To obtain negative partial product terms *{-4A, -3A, -2A, -A}* we have to take 2's complement of respective positive partial product values. The 2's complement is done in two steps: in the first step the 1's complement of *{4A, 3A, 2A, A}* is obtained and in second step to obtain negative terms, sign bit (n_i) is added to corresponding partial product term.

Radix-8 Booth algorithm has two parts: booth encoder and booth selector. The Radix8 booth encoder generates the partial product bits given in Table 1. The booth selector chooses one value from the partial product set *{A, 2A, 3A, 4A}* where the product term is represented in normal or complemented form. It uses control signals *{* w_i, x_i, y_i, z_i *}* to choose partial product set and use control bit (n_i) to produce partial product term for 1's complement form. The control signal of respective bit becomes active:

w_i *= 1* (becomes active) when partial product term *P = {A,-A};*

x_i *= 1* (becomes active) when partial product term *P = {2A,-2A};*

y_i *= 1* (becomes active) when partial product term *P = {3A,-3A};*

z_i *= 1* (becomes active) when partial product term *P = {4A,-4A};*

Radix 8 Booth multiplier structure for 16-bit is based on Partial Product Array (PPA) as shown in Fig. 1. To obtain the correct multiplication, for sign extension up to *(2n-1)* bit positions, the partial product row is shifted to left by 3-bit positions with respective to previous row to extend the sign. Thereby, there introduces an adder unit complexity, to compensate the problem introduce guard bits in every partial product row. In each row of the PPA the control bit *(n_i)* is added at the starting of each row to convert 1's complement number to 2's complement number and by this size for an extra row gets increased, so a carry bit *(e_i)* is generated to compensate the control bit.

Table 1. Partial product values for radix8 booth encoder

$b_{3i+2}b_{3i+1}b_{3i}b_{3i-1}$	Operation	w_i	x_i	y_i	z_i	n_i	pp_{ij}
0000	0	0	0	0	0	0	0
0001	+A	1	0	0	0	0	x_j
0010	+A	1	0	0	0	0	x_j
0011	+2A	0	1	0	0	0	x_{j-1}
0100	+2A	0	1	0	0	0	x_{j-1}
0101	+3A	0	0	1	0	0	$x_j + x_{j-1}$
0110	+3A	0	0	1	0	0	$x_j + x_{j-1}$
0111	+4A	0	0	0	1	0	x_{j-2}
1000	-4A	0	0	0	1	1	$\overline{x_{j-2}}$
1001	-3A	0	0	1	0	1	$\overline{x_j + x_{j-1}}$
1010	-3A	0	0	1	0	1	$\overline{x_j + x_{j-1}}$
1011	-2A	0	1	0	0	1	$\overline{x_{j-1}}$
1100	-2A	0	1	0	0	1	$\overline{x_{j-1}}$
1101	-A	1	0	0	0	1	$\overline{x_j}$
1110	-A	1	0	0	0	1	$\overline{x_j}$
1111	0	0	0	0	0	0	0

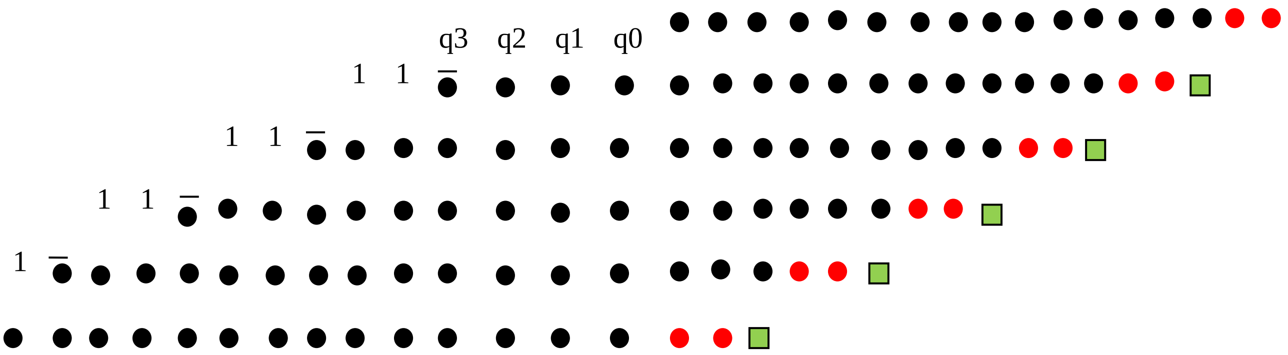


Figure. 1 Radix-8 16-bit partial product terms are shown in dot representation 🟩 is the absorbed carry bit of the recording factor; 🔴 are the modified bits by compensating the extra row; q0-q3 are the sign extension bits on modifying sign bit; ● a partial product term; 1 is the guard bit.

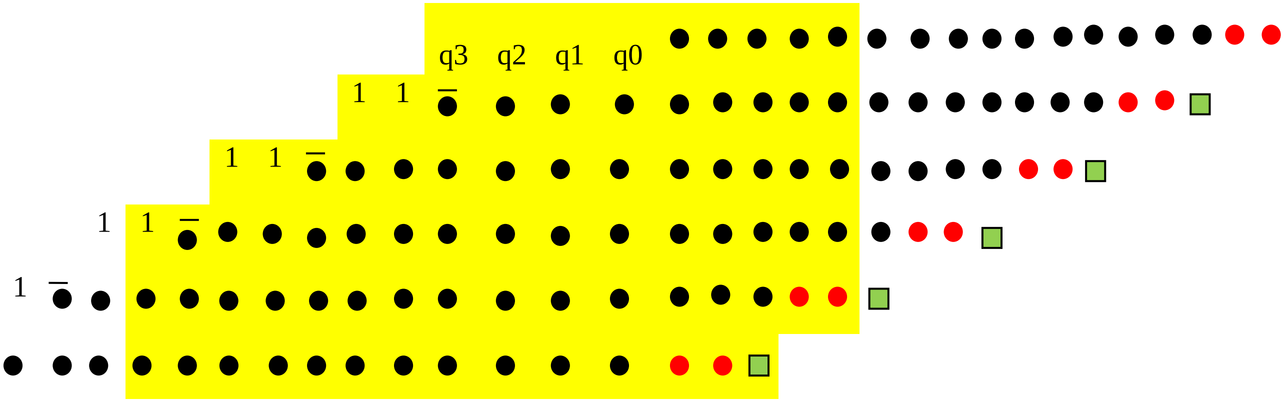
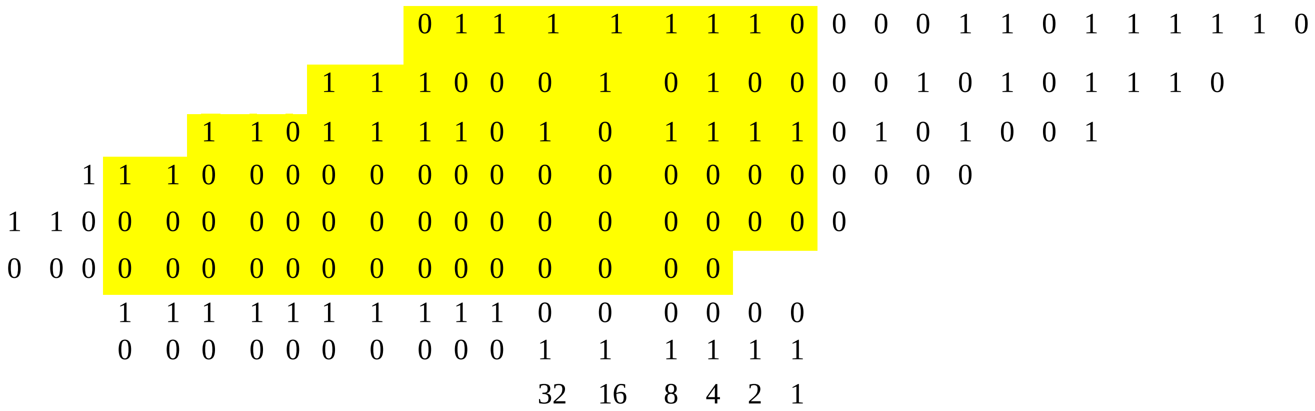


Figure. 2 Proposed method for 16-bit Radix8 Booth multiplier partial product terms



```
                    0 1 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 1 0
                1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0
          1 1 0 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 1
    1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
      1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
                          32 16 8 4 2 1
```

Figure. 3 Example showing the proposed multiplier for the coefficient (*1/α = -0.630464*)

The two LSB's of each row are modified by compensating the control bit and produce the carry bit. The sign bits can be generated by the modifying the sign extension bits where modified bit expressions are shown below:

$$m_{i0} = p_{i0} \wedge n_i \tag{5}$$

$$m_{i1} = p_{i1} \wedge (p_{i0} \& n_i) \tag{6}$$

$$e_i = p_{i,1} p_{i,0} n_i \qquad 0 \le i \le \frac{n}{3} - 2 \tag{7}$$

$$q0 = p_{0,15} \wedge r_5 \tag{8}$$

$$q3 = \overline{p_{0,15}\overline{r_5}} \tag{9}$$

$$q1, q2 = \overline{q3} \tag{10}$$

We have extended the approach given in [5] for 16-bit Radix-8 Booth multiplier and modified the PPA by only generating the required partial products. Generally in many VLSI applications the main aim is to achieve low area and low power with high performance which will impact the system level functionality. In the proposed method only the required product terms are generated and remaining lower and higher part is truncated, so that the number of generation terms gets reduced thus area gets reduced. Upon truncating the number of partial product terms and utilization of adder units gets reduced, so that speed is improved. The truncation used in this method is not uniform scaling because in this method we try to reduce the generation which are on the right side. By truncating the remaining terms a truncation error of *+/- 1%* error is introduced, which can be neglected for image processing applications.

In proposed method when each sample is multiplied with a constant, the multiplier two operands are of 16-bit binary numbers and the resultant output is of 32-bit number. In the proposed method the truncation done to both higher and lower order bits and we obtain a 16 bit number on truncation as shown in Fig. 2. This truncation is done on scaling coefficient value. Considering an example using the coefficient used in 2D DWT where (*1/α*) = *-0.630464*, the steps followed are shown below:

Step 1: Determine the scaling range for the coefficient.

Step 2: Determine the required partial product terms that should be generated from the booth encoder.

Step 3: Partial product terms are added using the carry save adder. The required scaling terms are to be taken as the output.

Step 4: In the above example the output is taken from 12th bit to 27th bit as shown in Fig. 2, the shaded part represents the desired output.

An example is shown on how the proposed multiplier is done is shown in Fig. 3 where X is the multiplicand (*X= -0.630464*) and Y is the multiplier (*Y= 99*). By proposed Booth multiplier the required product terms are generated (i.e12th bit to 27th bit) from the booth encoder. For the calculation of output, the entire column of the respective bits are added using the carry save adder. When multiplying X with Y the obtained output is -62.415 and by using proposed Booth multiplier the theoretically obtained output is -62 which is shown in Fig. 3. The truncation done for various coefficients used for 2D DWT is not uniform scaling because we try to scale the product terms generation part, so that in the booth encoder the number of partial products gets reduced as shown in Fig. 2. The various coefficients used in 2D DWT architecture their value, precision value; decimal and binary values are listed in Table 2. The below table shows how the scaling should be done for various coefficients. By using truncation technique the need for number of adder units, generation of partial terms gets reduced and ease of taking the output from the precision bit is easy.

Table 2. Precision values of DWT coefficients

Coefficient	Value	Precision	Decimal Value ($X2^{-precision}$)	Binary Value
(1/α)	-0.630464	12-bit	-2582	1111010111101010
(1/αβ)	11.900004	9-bit	6093	0001011111001100
((1/δγ)+1)	3.553775	11-bit	7278	0001110001101110
(1/βγ)	-21.37815	8-bit	-5473	111010101010000
((1/αβ)+1)	12.900004	9-bit	6605	0001100111001100

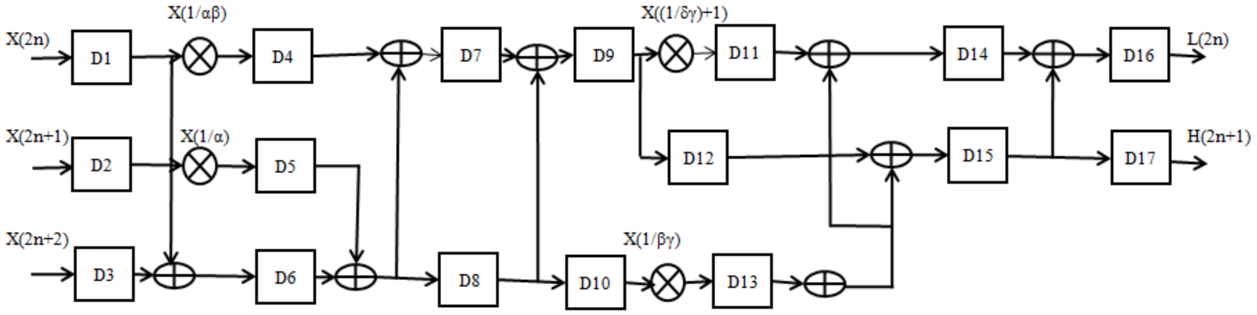


Figure. 4 Column filter

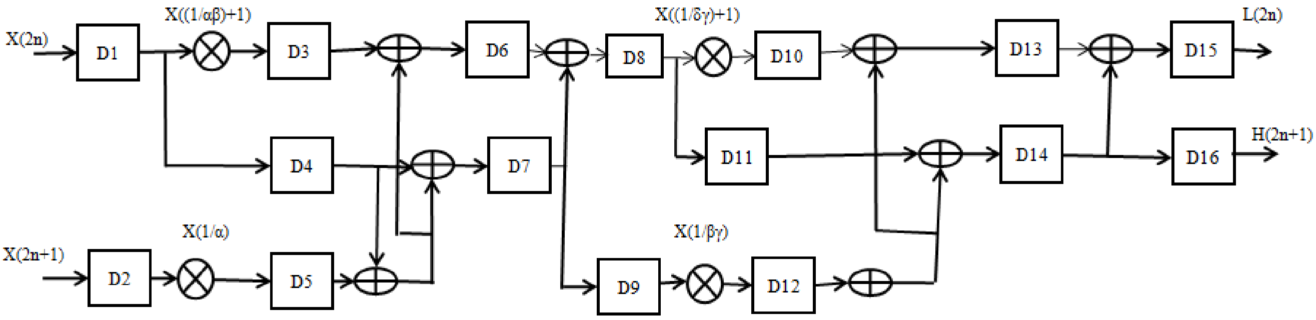


Figure. 5 Row filter

4. Proposed 2D DWT architecture

Normally in the existing architecture [11] the 1D DWT architecture consumes 12 clock cycles and the proposed architecture consumes 8 clock cycles for the operation by removing the extra 4 registers. By reducing the number of registers speed is increased and thus enhances the performance.

In the proposed 2D DWT architecture it consists of column filter, transposing unit and row filter. The column filters are of three inputs whereas row filter is two-input structure generating four sub-bands having two outputs each with reduced number of storage elements. Due to reduction of number of clock cycles for the proposed method the buffer size for memory is reduced and the image storage capability is increased. The optimized 1D column and row filter is shown in Figs. 4 and 5. When compared the proposed architecture with the lifting scheme [1] it occupies less number of clock cycles, so that speed is improved.

The total number of hardware components used in the 2D DWT is shown in Table 3. Below table comprises of the total number of registers, XOR/XNOR, AND/OR/NOR/NAND gates and latency required for the implementation of the 2DDWT architecture for CSD implementation, Booth multiplier [16] and proposed architecture. The CSD implementation consumes more number of registers and the proposed architecture requires less number of registers. This makes the advantage of acquiring less area for applying in various image processing applications. In CSD implementation the storage element (D flip flop) takes two cycles for the execution of the operation and to compensate the effect extra two registers are added in column filter using CSD implementation. But in proposed architecture for multipliers there is no need of registers so it will get executed with less number of clock cycles than CSD. The proposed 2D DWT architecture occupies less area by reducing a total of eight registers than in CSD implementation, by

Table 3. Hardware requirement for 2D DWT architecture

Method	Column filter				Row filter			
	XOR/ XNOR	AND/OR/ NOR/NAND	REG	LATENCY	XOR/ XNOR	AND/OR/ NOR/NAND	REG	LATENCY
CSD	768	1920	35	12	768	1920	35	12
[16]	1164	2536	17	8	1164	2536	16	8
Proposed	802	1949	17	8	802	1949	16	8

Table 4. Synthesis results of 2D DWT architecture

Method	Delay (ns)	Area (um^2)	Power (mW)	ADP	EADP (%)	PDP	EPDP (%)
CSD	1.522	42219.1	1.78	64257.5	29.0194	2.713187	26.12955
[16]	1.862	34297.4	1.56	63861.8	28.2248	2.903482	34.97588
Proposed	1.580	31521.9	1.36	49804.5	--	2.151112	--

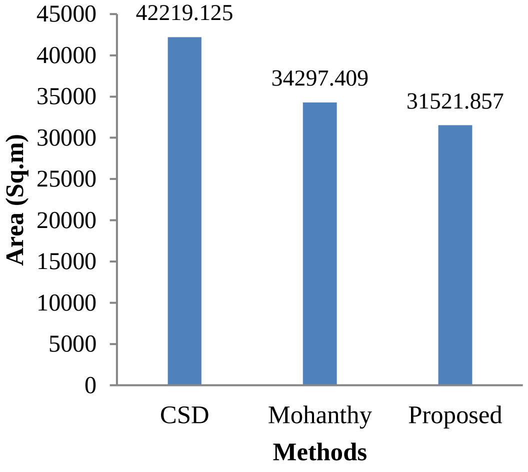


Figure. 6 Area comparisons of different methods

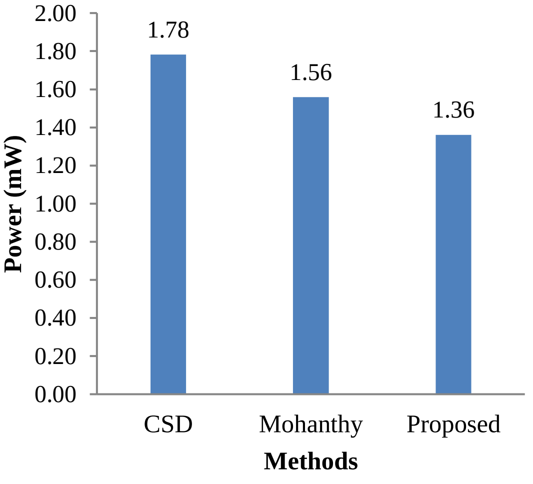


Figure. 7 Power comparisons of different methods

which the proposed architecture gains better performance. When comparing the latency of CSD and booth multiplier it occupies extra 2 clock cycles than the proposed multiplier.

5. Results

Design of the proposed work is done using Verilog HDL and the synthesis results are done in Cadence Genus of 90nm technology. The ASIC implementation results of proposed Radix8 Booth multiplier, CSD multiplier and Booth multiplier [16] are shown in Table 4 which shows parameters like delay, area and power. The ADP of the CSD architecture is 29.018% excess ADP (EADP) and for booth multiplier it is 28.22% EADP than proposed Booth multiplier and the PDP of the CSD architecture is 26.12% excess PDP (EPDP) and in booth multiplier it is 34.97% EPDP than proposed architecture are computed and it shows how much better performances does the proposed method gives than the existed method relating to area and power. The delay is not improved in proposed architecture than [11] because the critical path is considered from the truncated part. The throughput for the 2D DWT architecture in CSD based implementation has extra four cycles than the proposed architecture.

Table 4 shows in the proposed method the area and power is reduced when compared to [1,5] because all the partial product terms are not generated, only the required terms are generated so that area gets reduced. Figs. 6 and 7 show the graph for the different methods comparison based on area and power. These results are helpful for fast speed implementations and capable for picture handling applications. The simulation result for the proposed multiplier with (X=-0.630464, Y=99) is -63 when compared with the theoretical output has an error rate up to +/- 1%.

6. Conclusions

In this work, a 16-bit Radix-8 Booth multiplier is implemented and applied it to the 2D DWT architecture. The proposed architecture is compared with the CSD and observed that the area and power is reduced. Improvements in the architecture are done to occupy less number of clock duration cycles and to get a detailed output with no distraction of the value is done. On performing synthesis, the proposed method consumes less area and power, thus saving of 29.01% EADP and 26.12% EPDP for existing CSD based architecture and 22.25% EADP and 34.97% EPDP for existing booth multiplier is achieved with an error rate of +/-1%. This one level 2D-DWT architecture can be extended to multi-level 2D DWT as the future work to improve the design.

References

[1] C. T. Huang, P. C. Tseng, and L. G. Chen, "Flipping Structure: an efficient VLSI architecture for lifting-based discrete wavelet transform", *IEEE Tran. on Signal Processing*, Vol.52, No.4, pp.1080-1089, 2004.

[2] Y.K. Lai, L.F. Chen, and Y.C. Shih, "A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform", *IEEE Trans. Consum. Electron.*, Vol.55, No.2, pp.400-407, 2009.

[3] A. Darji, R. Arun, S. N. Merchant, and A. Chandorkar, "Multiplier-less pipeline architecture for lifting-based two-dimensional discrete wavelet transform", *IET Computers & Digital Techniques*, Vol.9, No.2, pp.113-123, 2014.

[4] P. K. Meher, B. K. Mohanty, and M. N. S. Swamy, "Low-Area and Low-Power Reconfigurable Architecture for Convolution-Based 1-D DWT Using 9/7 and 5/3 Filters", In: *Proc. of Int. Conf. VLSI Design*, pp.327-332, 2015.

[5] B. K. Mohanty and P.K. Meher, "Memory-efficient modular VLSI architecture for high-throughput and low-latency implementation of multilevel lifting 2-D DWT", *IEEE Trans. Signal Process.*, Vol.59, No.5, pp.2072-2084, 2011.

[6] Y. Hu and C.C. Jong, "A memory-efficient scalable architecture for lifting-based discrete wavelet transform", *IEEE Trans. Circuit Syst. II Express Brief*, Vol. 60, No. 8, pp. 502-506, 2013.

[7] B. K. Mohanty and P. K. Meher, "Memory efficient high speed convolution based generic structure for multilevel 2D DWT", *IEEE Trans. Circuits Syst. Video Technol.*, Vol.23, No. 2, pp. 353-363, 2013

[8] B.K. Mohanty and P.K. Meher, "Area-delay-power-efficient architecture for folded two-dimensional discrete wavelet transform by multiple lifting computation", *IET Image Process*, Vol.8, No.6, pp.345-353, 2014.

[9] S. Kuang, J. Wang, and C. Guo, "Modified Booth multiplier with a regular partial product array", *IEEE Trans. Circuits Syst. II*, Vol.56, No.5, pp.404-408, 2009.

[10] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high-performance operation", *IEEE Trans. Comput.*, Vol.65, No.8, pp.2638-2644, 2016.

[11] C. Wu, W. Zhang, and J. Liu, "Hardware efficient multiplier-less multi-level 2D DWT architecture without off-chip RAM", *IET Image Processing*, Vol.11, pp. 362-369, 2017.

[12] Y. K. Lai, L. F. Chen, and Y.C. Shih, "A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform", *IEEE Trans. Consum. Electron.*, Vol. 55, No. 2, pp. 400-407, 2009.

[13] B. K. Mohanty, A. Mahajan, and P.K. Meher, "Area and power-efficient architecture for high-throughput implementation of lifting 2-D DWT", *IEEE Trans. Circuits Syst. II Express Briefs*, Vol. 59, No. 7, pp.434-438, 2012.

[14] A. Darji, S. Agrawal, and A. Oza, "Dual-scan parallel flipping architecture for a lifting-based 2-D discrete wavelet transform", *IEEE Trans. Circuit Syst. II Express Brief*, Vol.61, No.6, pp.433-437, 2014.

[15] W. Zhang, Z. Jiang, Z. Gao, and Y. Liu, "An efficient VLSI architecture for lifting-based discrete wavelet transform", *IEEE Trans.Circuits Syst. II Express Briefs*, Vol.59, No.3, pp.158-162, 2012.

[16] B. K. Mohanty and A. Choubey, "Efficient design for radix-8 booth multiplier and its application in lifting 2-D DWT", *Circuits Syst. Signal Process.*, Vol.36, No.3, pp.1129–1149, 2017.