EMITTER International Journal of Engineering Technology
Vol. 13, No. 2, December 2025, pp. 333~351
DOI : 10.24003/emitter.v13i2.819

Design of Optimized Hardware Architecture for Discrete Cosine Transform using Loeffler's Algorithm

Chinmayi S V[1], Veena M B[2]

[1,2] Department of Electronics and Communication Engineering, BMS College of Engineering, Bangalore, India
Corresponding Author: chinmayisv.lel21@bmsce.ac.in

Received September 14, 2025; Revised October 26, 2025; Accepted December 2, 2025

Abstract

The Discrete Cosine Transform (DCT) is the most commonly used transformation technique in image processing applications for data compression. It transforms a finite set of data points or pixels into frequency domain in terms of sum of cosine coefficients of various frequencies. This paper proposes an optimized hardware architecture for 2D 8x8 Discrete and Inverse discrete cosine transforms (DCT and IDCT) using Loeffler's algorithm. The hardware architecture is optimized using efficient adders and multipliers. Modified carry select adders (CSLA) are used to boost the speed, wherein Booth multipliers improve overall performance of the design. The Loeffler's algorithm consisting of 8-stage pipelined architecture reduces the arithmetic operations per cycle and improves processor efficiency. The front-end RTL design of the proposed architecture is implemented on Virtex-7 FPGA, while the backend design is implemented in Cadence using 45nm CMOS technology. The proposed design possesses 24% lesser area, 25% lesser leakage power and 8.8% lesser delay than the existing designs.

Keywords: Discrete Cosine Transform, Inverse Discrete Cosine Transform, DCT, IDCT, Loeffler algorithm.

1. INTRODUCTION

Portable multimedia devices have evolved largely in terms of features and technology. From portable cassettes, CD players and MP3 players to the most sophisticated smart phones, video players, tablets, streaming services like YouTube, Netflix etc, the evolution of multimedia also demands for advancement in image and video processing techniques, hence requiring sophisticated and efficient compression techniques [1]. Also, there is a need to optimize the hardware design of compression techniques. DCT is one such image compression technique widely used in video and image processing applications. Various DCT architectures have been proposed in previous studies as discussed in section 2 below. In this paper, an optimized hardware architecture is proposed for DCT using 8-stage pipelined Loeffler's algorithm with only 11 multipliers and 29 adders as the conventional DCT design

requires 64 multiplications and 56 additions for N=8 point 1D DCT/IDCT. The Loeffler's architecture is discussed in section 4 below. High speed carry select adders are used to boost the speed of the design. Same adders are used also as subtractors for reducing design complexity. This adder design is further delay-optimized by breaking the lengthier ripple carry adders into smaller blocks that perform parallel processing, to avoid delay due to carry propagation. To improve the overall performance in terms of power, area and speed, booth multipliers are used in the proposed work as the shift multipliers used in existing designs are inefficient in terms of hardware utilization and power consumption. The front-end RTL design is carried out on Virtex-7 FPGA and in backend using CMOS 45nm technology in Cadence to measure the power, delay and area of the design on the hardware IC and generate physical layout of the design.

2. RELATED WORKS

Various papers have proposed different implementations for DCT architecture. DCT is a lossy compression technique extensively used for video and image compression in standard algorithms such as JPEG, MPEG, H.26x etc. Due to its energy compaction property, DCT is also used for image compression in real time applications like wireless capsule endoscopy [2][3]. A CMOS based approach is realized using Cordic algorithm and a multiplier-less approach [4]. But this design lacks accuracy as it uses approximate adders and optimization and rounding-off of constant factors. The High Efficiency Video Coding HEVC [5]-[8] is a new standard that also used DCT for compression. An efficient DCT design proposed for HEVC compliant devices [9][10] uses integer conversion to achieve half the bit rate needed for H.264 standard. This design comes with lesser area and energy requirements but has more design complexity. To perform power and area optimization for DCT design for HEVC standard, a muxed MCM based approach [11][12] was proposed, which reduced the hardware cost by around 30%. There are also other DCT architectures using Taylor-series expansion [13]-[15].

Since the conventional DCT computation required 64 multiplications and 56 additions to calculate N=8 point 1D DCT, Loeffler proposed an algorithm [16] that requires only 11 multiplications and 29 additions which is the least theoretical limit possible till date. A hardware accelerator for DCT using improved 8-stage pipelined Loeffler's algorithm [17] uses multiplier-less design with shift and add multipliers and this design includes iterative shifting and addition operations, causing area overhead, as it consumes large number of logic blocks on the FPGA. It also consumes more power and has higher delay. Also, the adders used are ripple carry adders (RCA) which come with larger delay due to rippling of carry from 1st stage to further stages.

3. ORIGINALITY

This paper aims at designing an optimized hardware architecture for Discrete Cosine Transform (DCT) and Inverse DCT using improved 8-stage pipelined Loeffler's architecture. The optimization is achieved using high-speed carry select adders (CSLA) with BEC for reducing delay and Booth multipliers for area and power optimization. The adder design is further optimized to reduce carry propagation time by dividing lengthier ripple carry adders in the CSLA into smaller blocks that are processed parallely. Same adders are used as subtractors to reduce design complexity. The improved Loeffler's architecture used in the design requires least possible number of adders and multipliers. It divides the number of arithmetic operations evenly across all clock cycles and also improves processor efficiency. The architecture is implemented in both front end and back end. The front-end RTL design is carried out using Xilinx ISE on Virtex-7 FPGA and the backend design uses Cadence RTL compiler and Encounter softwares to generate design layout in CMOS 45nm technology.

4. SYSTEM DESIGN

4.1 Loeffler's Algorithm

Considering a original input image x(i, j) and transformed output image y(u, v). The equation to compute 1D DCT and IDCT for an N-point sequence are given by,

$$DCT, y(u) = \sqrt{\frac{2}{N}} c(u) \sum_{i=0}^{N-1} x(i) \cos \frac{(2i+1)u\pi}{2N} \quad (1)$$

$$IDCT, x(i) = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} c(u) y(u) \cos \frac{(2i+1)u\pi}{2N} \quad (2)$$

Where, x(i) is the original input pixel values along a row or column and y(u) is the transformed output pixel values along the corresponding row or column. c(u) is the constant factor, where c(u)=0.707u for v=0 and c(u)=u for v=1,2,….N-1. From (1) and (2), for a 1D DCT and IDCT with N=8, it takes 56 additions and 64 multiplications. By using the Loeffler's algorithm [10], 1D DCT and IDCT requires only 11 multiplications and 29 additions. Figure 1 below shows the butterfly structure of Loeffler's architecture for 1D DCT for N=8. Figure 2 presents the symbolic operations of the butterfly structure.

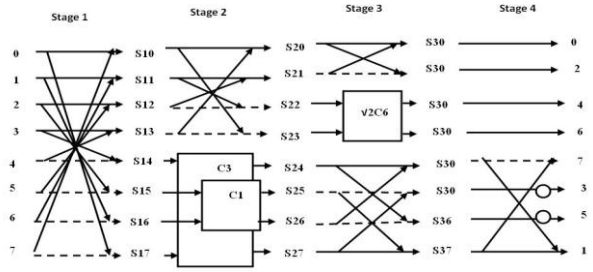

Figure 1. Loeffler algorithm for 8-point DCT.

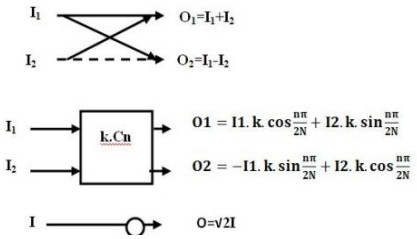

Figure 2. Symbolic operations of the algorithm

But, in this architecture, the resources (adders and multipliers) are distributed unevenly in all clock cycles. More resources are on the critical path which adds to the delay of the circuit. Hence in [17], the Loeffler algorithm is improvised as an 8-stage pipelined architecture and operations are distributed evenly, with only one addition or one multiplication at every step. This helps to increase the operating frequency and improves processor efficiency and cycle-time utilization. Table 1 and 2 below summarizes the 8-stage pipelined architecture of Improved Loeffler 1D DCT and IDCT for N=8. Table 3 consists of cosine constant factors for N=8 DCT/IDCT.

Table 1. Improved 8-stage Loeffler algorithm for 1D DCT data stream

		Clock cycle								Output reg
		1	2	3	4	5	6	7	8	
Input reg	0	0+7	0+3	0+1						0
	1	1+6	1+2	0-1						4
	2	2+5	1-2					(c-f).3	8+2	2
	3	3+4	0-3					(c+f).2	8-3	6
	4	3-4	(e-d).7			8+4	4+6	7-4		7
	5	2-5			(g-b).6	9+5	7-5		5.h	3
	6	1-6			(g+b).5	9-6	4-6		6.h	5
	7	0-7	(e+d).7			8-7	7+5	7+4		1
Added reg	8		4+7	8.d		2+3	8.f			
	9		5+6	9.b						

Table 2. Improved 8-stage Loeffler algorithm for 1D IDCT data stream

		Clock cycle								Out put reg
		1	2	3	4	5	6	7	8	
Input reg	0->0						0+1	0+3	0+7	0
	4->1						0-1	1+2	1+6	1
	2->2		(c+f).3		8-2			1-2	2+5	2
	6->3		(c-f).2		8+3			0-3	3+4	3
	7->4		7-4	4+6	(e+d).7			8-4	3-4	4
	3->5	h.5		7-5		(g+b).6		9-5	2-5	5
	5->6	h.6		4-6		(g-b).5		9+6	1-6	6
	1->7		7+4	7+5	(e-d).4			8+7	0-7	7
Added reg	8		2+3	8.f	4+7		8.d			
	9					5+6	9.b			

Table 3. Corresponding Values of Constant Cosine Factors

a	**b**	**c**	**d**	**e**	**f**	**G**	**h**
$\cos\frac{\pi}{4}$	$\cos\frac{\pi}{16}$	$\sqrt{2}\cos\frac{\pi}{8}$	$\cos\frac{3\pi}{16}$	$\cos\frac{5\pi}{16}$	$\sqrt{2}\cos\frac{3\pi}{8}$	$\cos\frac{7\pi}{16}$	$\sqrt{2}$

The proposed design is an optimized hardware architecture for 8-point 1D DCT and IDCT using improved 8-stage Loeffler's architecture. The design makes use of 2 main logic blocks:

1. Improved 16-bit Carry select adder (CSLA) with Binary to Excess-1 Converter (BEC) for increasing the speed of operation.
2. Booth Multiplier for area and power optimization.

4.2 Improved 16-bit Carry select adder with BEC

Adders are the basic elements of any digital circuit that are used in various arithmetic operations. The generic ripple carry adders (RCA) used in the DCT design in [17] are the simplest and most basic adders. The disadvantage of RCA is that they have larger delay due to the time required to transport carry from every stage to its next stage, all along the adder [18]. Carry select adder is a high speed adder suitable for applications requiring fast processing of data [19]-[22]. The carry propagation delay is reduced in CSLA by generating two sums simultaneously, with carry_in=0 and carry_in=1 at every stage and then selecting the right sum on the basis of carry_out of previous stage using multiplexer [23]-[25]. However, the CSLA is not area efficient as it uses more pairs of basic adders for simultaneous computation. The conventional CSLA is as shown in Figure 3 below.

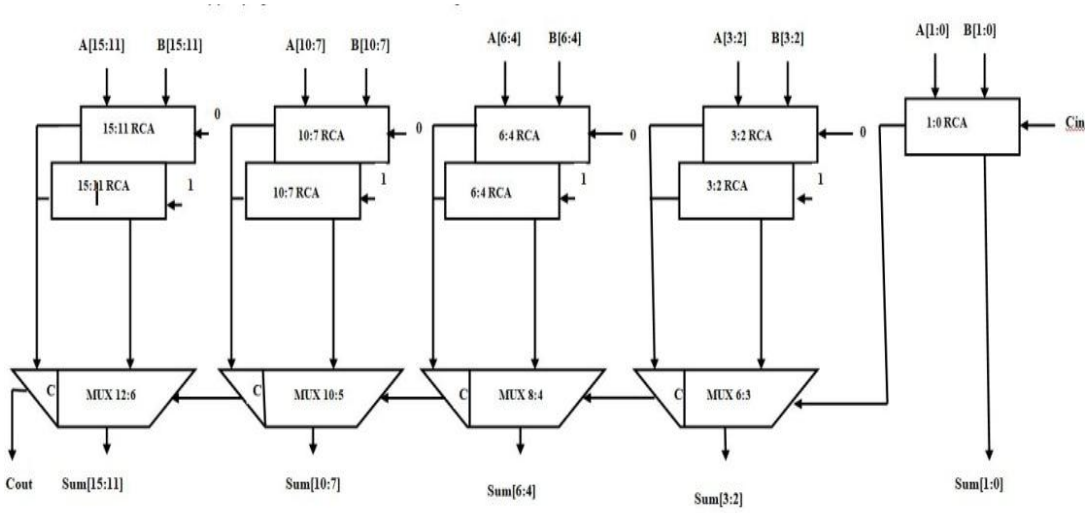

Figure 3. Conventional 16-bit CSLA with RCA

The conventional CSLA uses RCA sub-stages, which again adds on to the delay, but at a minor level. But, the binary to excess-1 converters does the job more efficiently with lesser delay and lesser hardware utilization compared to conventional CSLA [26]. The existing 16-bit CSLA with BEC is shown in Figure 4 below.

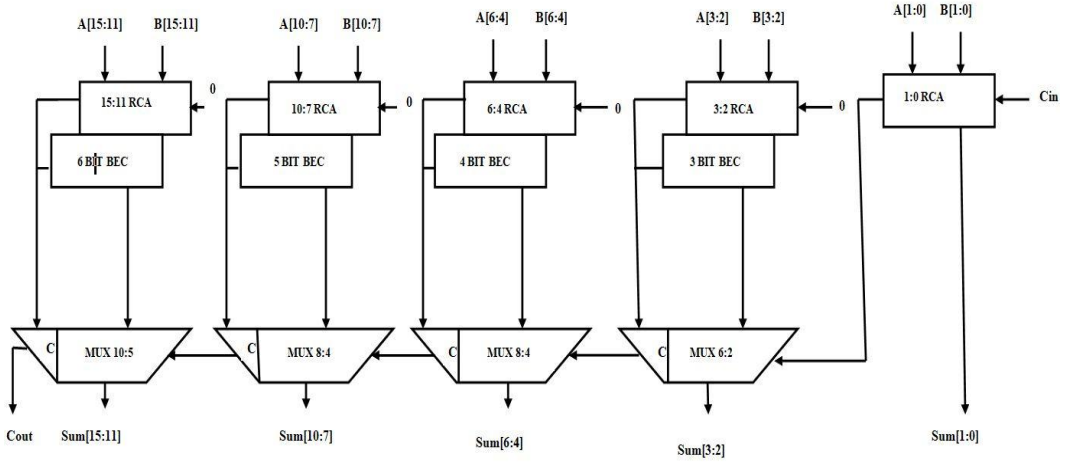

Figure 4. Existing 16-bit CSLA with BEC

Here, till the time sum and carry_out of first stage are available, the next stage ripple carry adder computes sum for higher bits of the operands with carry_in=0 parallely. The output of this RCA top stage is fed as input to BEC, also called add-one circuit, which adds one to the sum, equivalent to computing the sum of higher bits with carry_in=1 separately. Later, the carry_out available from previous stage is used as select line for the multiplexer to choose correct sum for higher order bits. This is a low power technique called pre-computation used for reducing the power consumption of combinational circuits [27].

The above shown 16-bit CSLA with BEC utilizes 4-bit and 5-bit ripple carry adders at fourth and fifth stages. In the proposed improved CSLA, these lengthier RCAs are split into smaller blocks in order to further speed up the process. The proposed 16-bit CSLA with BEC is as shown in Figure 5 below.

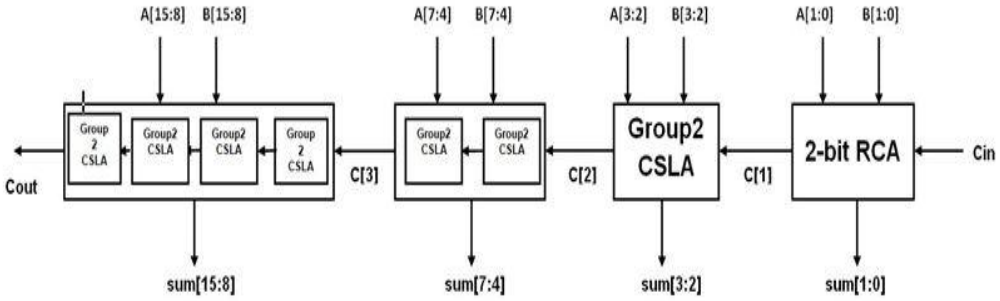

Figure 5. Proposed modified 16-bit CSLA with BEC

In this design, a block of 2-bit RCA and 3-bit BEC is used in the second stage to compute sum of next 2 bits and it is named as group-2. This block is used inside next stages to process higher order bits. The block diagram of Group-2 stage is as shown in Figure 6 below.

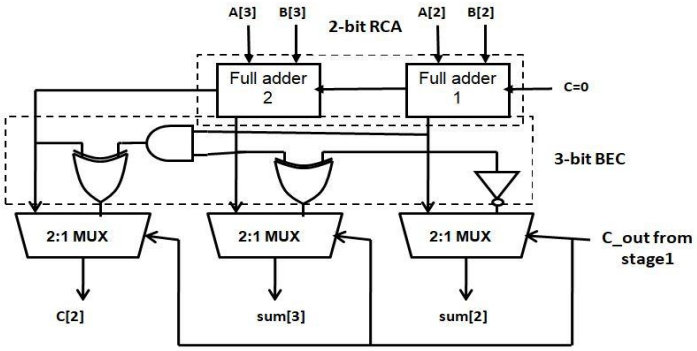

Figure 6. Group2 stage used in the proposed CSLA.

Considering sum1[2] and sum1[3] as sums from full adders 1 and 2 and c_1[2] is the carry out from full adder 2 fed as inputs to the 3-bit BEC, the BEC adds a value 1 to this sum. If sum2[2], sum2[3] and c_2[2] are the outputs of BEC, the equations for these outputs are given below in (3), (4) and (5). These outputs, along with sum1[2] and sum1[3] are given as inputs to MUX.

$$sum2[2] = \sim sum1[2] \tag{3}$$

$$sum2[3] = sum1[3] \wedge sum1[2] \tag{4}$$

$$c\_2[2] = (sum1[3] \,\&\, sum1[2]) \wedge c\_1[2] \tag{5}$$

The third stage of CSLA calculates sum of next 4 bits and uses two group-2 blocks parallely. The fourth stage calculates sum of higher order 8 bits and uses four group-2 blocks. By doing this, the critical paths are split, and delay is reduced further. This design also reduces one stage compared to Figure 3.

Each Group2 CSLA block in the proposed design has a delay of approximately 9ns. Since the computation of sum happens parallely across all four stages in advance for C_in=0 and C_in=1, not much time is wasted in carry propagation, making it a faster adder design for real-time computations compared to the conventional adders.

Table 4. Comparison between conventional RCA and proposed adder with CSLA

Adder design	Delay in ns	Performance efficiency
Conventional RCA	15.27	
Proposed optimized adder with CSLA	11.10	27.3%

4.3 16-bit Booth Multiplier

Multipliers are the most important building blocks of any digital system. High performance multipliers are the need of any modern DSP system. In the design of hardware architecture for DCT in [17], a multiplier-less architecture is implemented using shift and adds multipliers. Shift multiplier is the most basic binary multiplier and is similar to paper and pen method of multiplication. For two n-bit operands, it requires n-stages of shift and add operations. While it is easy and straightforward to implement in hardware, it is not an efficient choice of multiplier, as it consumes more number of logic blocks and has more delay compared to other binary multipliers. The proposed design in this paper uses 16-bit booth multipliers. Binary multipliers are more efficient compared to the Vedic, Dadda, Wallace-tree and Array multipliers. Table 5 below presents a comparison of various types of multipliers as in [28].

Table 5. Comparison between different multipliers

	Vedic multiplier	Dadda multiplier	Wallace-tree multiplier	Booth multiplier
Area (LUTs)	1188	350	350	110
Delay (ns)	31.526	48.235	48.235	6.152
Power (mW)	0.274	0.275	0.274	0.277

The traditional method of multiplying two binary numbers, also called “long multiplication” is very time-consuming and requires more adder stages. The Booth multiplication algorithm brings down the number of partial products and additions, making it more efficient for hardware implementation and reduces the complexity of the circuit [29]. Figure 7

below shows the block diagram of booth multiplier. The booth algorithm works by splitting the binary number multiplication into smaller steps. It takes advantage of the repetitive patterns of 1s and 0s in the multiplier to generate a series of partial products that can be combined to get the result.

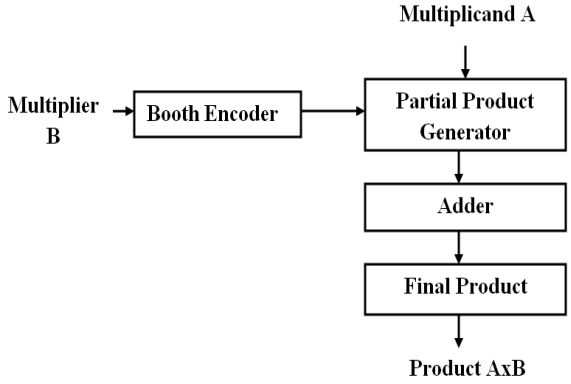

Figure 7. Booth multipliers

The steps of booth algorithms are:

1. Pre-processing: To extend the bit lengths of multiplicand and multiplier to handle 2's complement representation.
2. Booth Encoding: It is used to simplify multiplication by grouping adjacent 1s and 0s.
3. Partial product generation: Booth encoded bits are used to create sets of partial products. These are either multiplicand, shifted version of multiplicand or zero.
4. Partial product addition: Adding all the partial products together to get final result.

Table 6 below shows the performance comparison of shift multiplier and booth multiplier, from which it is observed that the booth multiplier is efficient compared to the former.

Table 6. Comparison between 16-Bit Shift Multiplier and Booth Multiplier

	16-bit shift multiplier	**16-bit booth multiplier**
Area (LUTs)	382	240
Power	33.284W	32.504W
Delay	15.27ns	11.1ns

4.4 Proposed DCT architecture

Figure 8 below shows the 8-stage Loeffler based architecture for 1D DCT for N=8. In this 8-stage pipelined structure, the critical path delay is reduced by evenly distributing the arithmetic operations across all the 8 stages. Each stage register performs only one arithmetic operation per clock cycle ensuring that there is no bottleneck at any stage. This balanced distribution of operations helps in increasing the cycle-time utilization and operating frequency. It also helps to prevent resource congestion and critical

path delays and thus balances the computation and data transfer in an optimal way.

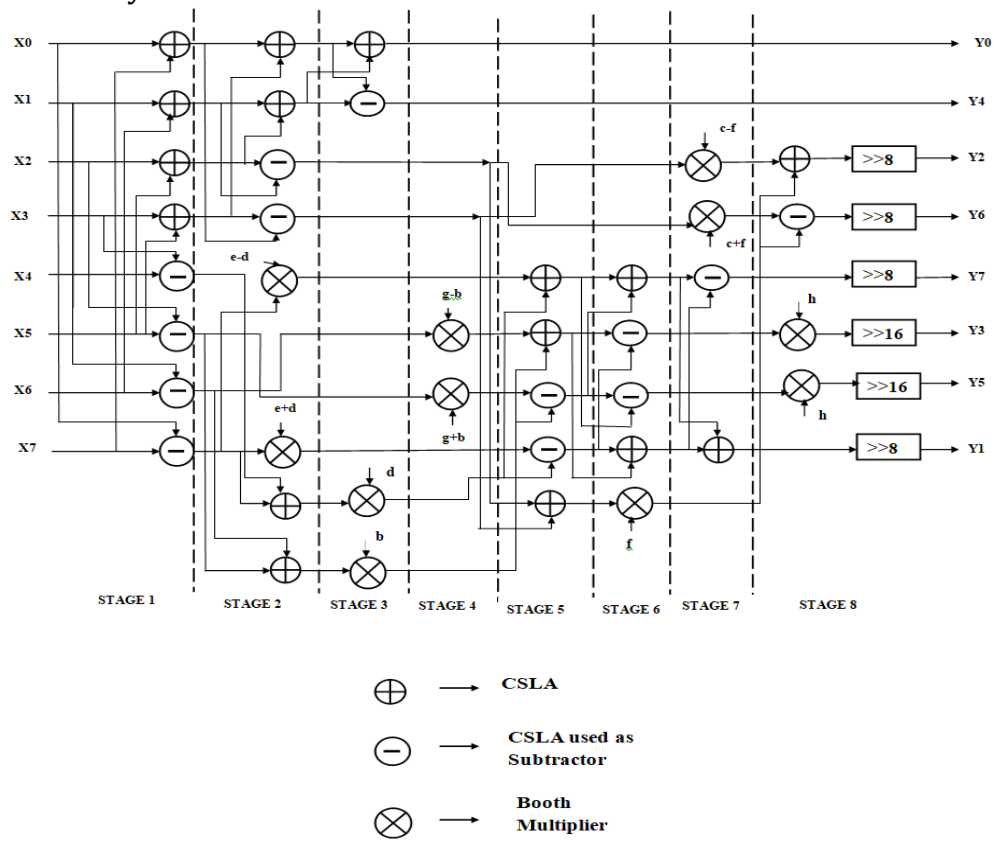

Figure 8. Proposed 1D DCT architecture

Once corresponding block operations at each stage are performed, the intermediate values are stored in next stage registers. For example, the inputs x0-x7 are added and subtracted and stored in next stage registers. In the proposed design, 16-bit carry select adders are used for addition and same are used as subtractors at various stages. The multiplication with constant factors is performed using 16-bit booth multipliers.

The inputs to the design are RGB pixel values ranging from 0 to 255, hence x0-x7 are taken as 8-bit variables. The DCT transformed outputs y0-y7 are considered as 14-bit signed integers. Zero padding, bit adjustments and possible truncations are done at intermediate stages. Since the proposed design is for fixed point integers and the constant cosine factors are floating point values, the constants are expresses in terms of coefficients of 256. That means, the constant values are multiplied by 256 and then rounded off to nearest integer for easier processing. Later, shifting to right by 8 bits is performed to obtain original transformed output values. This rounding-off of the floating-point values to fixed integers introduces very minor quantization errors that are negligible for practical applications. This method also simplifies the hardware implementation while preserving sufficient accuracy with minimal errors when reconstructed using IDCT, hence not creating any

significant negative impact on the visual fidelity. Table 7 below shows the floating to fixed integer conversion of constant cosine factors.

Table 7. Expressing Cosine terms as Fixed Integers

Constant cosine factor	Expressed as coefficient of 256	Binary representation
e-d	-71	111110111001
e+d	355	000101100011
d	213	000011010101
b	251	000011111011
g-b	-201	111100110111
g+b	301	000100101101
f	139	000010001011
c-f	196	000011000100
c+f	473	000111011001
h	362	000101101010

4.5 Proposed IDCT architecture

Figure 9 below presents the proposed hardware architecture for an 8-point IDCT using the same Loeffler architecture as in DCT. The IDCT is the reverse process of DCT for reconstruction of original image[30]. The 1D IDCT also consists of 11 multiplications and 29 additions. The inputs to IDCT are 14-bit transformed DCT values y0-y7, but swapping of register positions is needed as per the Loeffler algorithm for IDCT as in Table II. The output of IDCT is untransformed 8-bit unsigned pixel values x0-x7 chosen for DCT. In Loeffler 2D IDCT architecture, the output values are enlarged by 8 times which is why shifters are used to adjust the values at intermediate and final stages.

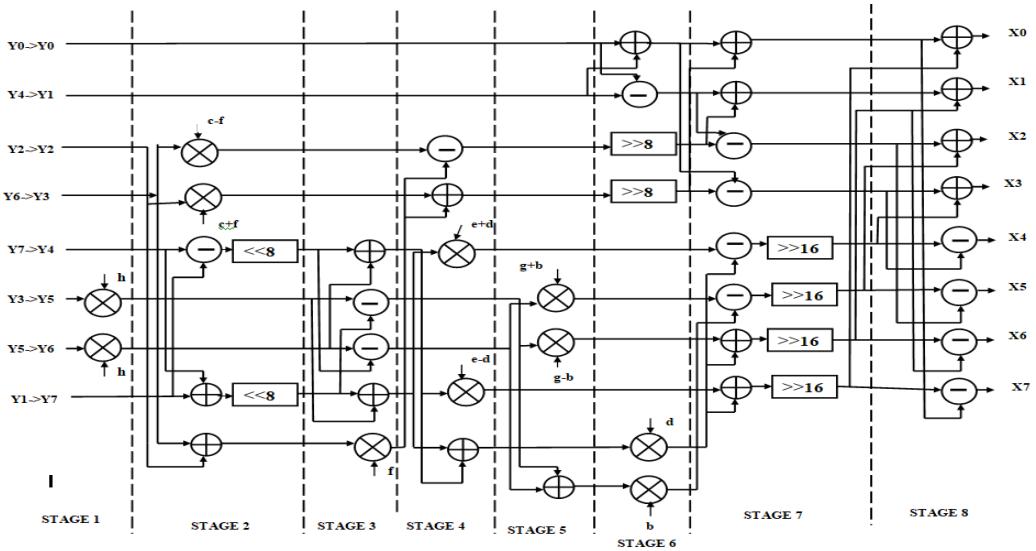

Figure 9. Proposed 1D IDCT architecture

The computational complexity is less in the proposed DCT design as it uses only 11 multipliers, 26 adders and 6 shifters in comparison with 51 adders and 34 shifters used in the existing design [12]. The 2D DCT/IDCT are obtained as per below Figure 10. The original image is split into blocks of 8x8 pixels to compute 2D DCT. The 2D computation is split into eight sequential 1D computations row by row. It takes 8 clock cycles to perform 1D DCT/IDCT computation and once clock count reaches 8, next set of 8 values are fetched and processed.

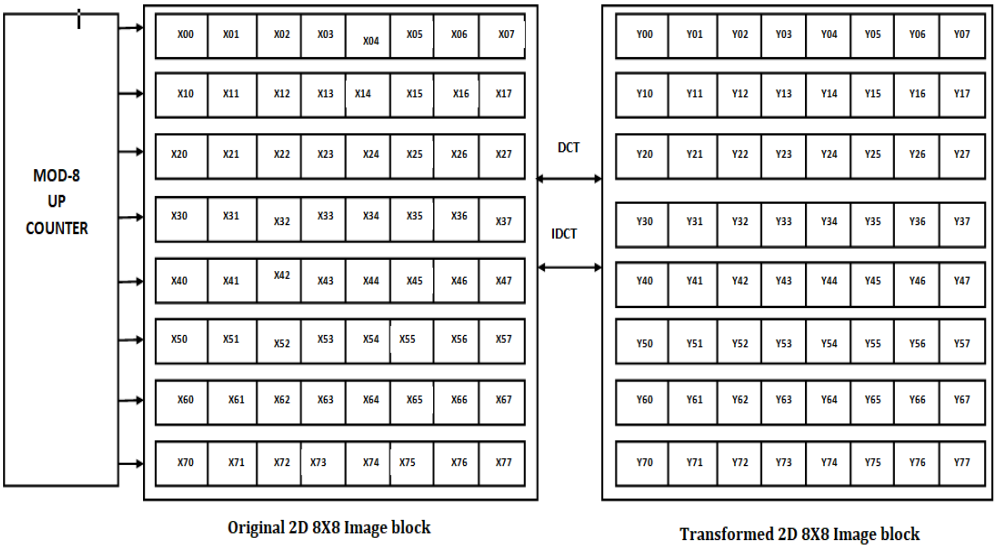

Figure 10. 2D DCT/IDCT implementation

A mod-8 up counter is implemented to count the 1D DCT transformations. Once all 8 rows are transformed, column-wise transformation is done on the row transformed values. Once the count value again reaches maximum limit, a control signal 'done' is set to mark the completion of 2D DCT computation. Similarly, 2D IDCT is also implemented to perform inverse operation for reconstructing the original pixel values.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Front End RTL Design

The proposed design is implemented in Xilinx ISE 14.7 platform using verilog HDL on Virtex-7 FPGA device. The existing design in [17] is also re-implemented for comparing its performance against the proposed design. The proposed DCT design is simulated, and the simulation waveforms are shown in Figure 11 below.

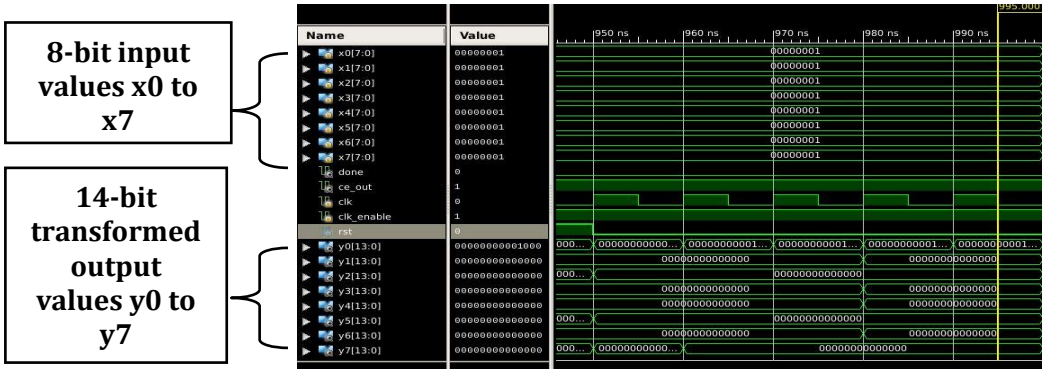

Figure 11. Simulation waveforms of proposed DCT architecture.

The inputs to the DCT design are 8-bit pixel values x0-x7 and the output data width is considered to be 14 bits transformed values. Control path in the design is implemented with control signals such as clk, count, reset, done etc, which are also shown in the waveform. The design is run for various input combinations are results are validated by computing inverse DCT. The simulation waveforms of the proposed inverse DCT are as shown in Figure 12 below.

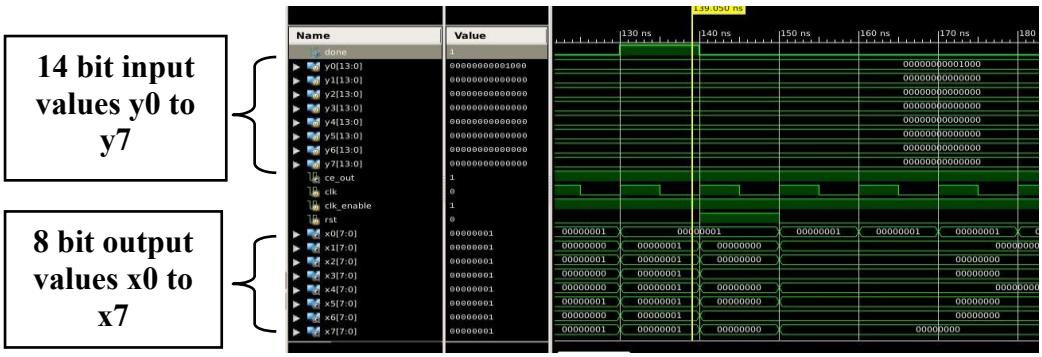

Figure 12. Simulation waveforms of proposed IDCT architecture.

The inputs to IDCT are 14-bit signed transformed values y0-y7 and output of IDCT gives original untransformed values x0-x7. The existing and proposed designs are synthesized using Xilinx ISE 14.7 and the device utilization summary is obtained. Table 8 presents the comparison between existing and proposed DCT and IDCT designs in terms of number of LUTs, slice registers and flipflops. From the below Table, it is observed that the proposed DCT design consumes 448 LUTs, and 73 slice registers lesser than the existing design. Thus, the proposed DCT design is optimized to consume 22% lesser area than the existing architecture on an FPGA.

Table 8. Comparison of device utilization on Virtex-7 FPGA.

	Existing architecture [17]		**Proposed architecture**	
	DCT	**IDCT**	**DCT**	**IDCT**
No of slice registers	222	242	149	153
No of slice LUTs	2059	2180	1611	1674
No of fully used LUT-FF pairs	198	198	146	146
No of DSP48E1s	6	6	5	5

5.2 Back End Design

VLSI back-end design refers to the process of designing and implementing the physical layout of the IC on silicon wafer. It is the phase of VLSI design which is followed after the front-end RTL design. This phase involves converting the logical representation of the design into physical layout. It determines the performance, shape and size of the IC. The back-end design of the existing and proposed hardware architectures is carried out using Cadence RTL Compiler v12 and Cadence Encounter Suite. Steps involved in the backend design of proposed DCT/IDCT architectures are as follows.

- Cadence RTL compiler is used to synthesize the verilog top module by running a TCL script to include CMOS 45nm standard cell library. Once synthesized, area, power and timing reports are generated.
- The compiler output is a structural verilog file instantiated with the CMOS standard cell library, used in the generation of design layout using Cadence encounter. In Encounter suite, the structural code and LEF file from standard cell library are imported and automatically, a core area is allocated for the design. Floor planning is done by first allocating space for power lines Vdd and Vss. Power planning is performed by drawing power rings with 6µm width each for Vdd and Vss and 1.5µm spacing between each line. Horizontal and vertical power strips are also added with 3µm spacing between each power strip for powering up the transistors.
- Placement of standard cells is done into the core area and routing is performed according to the structural code file.

The physical layout generated for DCT using Cadence is as shown in Figure 13 below. Similar layout is generated for IDCT following same steps. It consists of silicon substrate on which basic building blocks- transistors are placed, metal layer assignment for interconnecting various components, routing and interconnections to create path for signal flow between the transistors, stable power and ground distribution in rings and grids for powering up the IC.

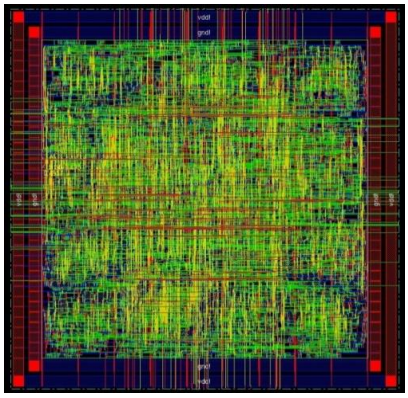

Figure 13. Physical layout of proposed DCT architecture

A comparative analysis is carried out to measure the performance of proposed design with the existing one. Reports are generated for area, delay and power of the circuits. Area comparison is carried out based on total number of cells and the overall area in μm^2. Timing is measured in terms of worst-case delay of the circuit and power consumption is with respect to the leakage and switching power. It is shown that the proposed hardware architecture is an efficient and optimized architecture in terms of all three metrics. It requires 24% lesser area on IC, dissipates 25% lesser leakage power and has a reduced worst-case delay of 8.8%, while the power required for switching activity almost remains the same. Table 9 provides a comparison of performance of the backend design.

Table 9. Performance comparison of the Backend Design

Performance metrics		**Exisitng design in [17]**	**Proposed design**	
			DCT	**IDCT**
Area	**No of cells**	7835	6604	6727
	μm^2	362	273	278
Leakage power	**nW**	677	507	520
Delay	**Ns**	91	83	84

6. CONCLUSION

This paper presents an optimized hardware architecture for DCT/IDCT using improved Loeffler's algorithm and the combination of carry select adders and booth multipliers. The proposed design is implemented in both front end and back end, and it is shown from the experimental results that the proposed work is optimized in terms of area, delay and power by 24%, 8.8% and 25% respectively compared to the existing architecture. The delay optimization is achieved with the help of modified high-speed carry select adders, while booth multipliers improve overall performance of the system. This improvement directly translates into faster image compression and decompression in multimedia applications while preserving high image quality. When applied to hyperspectral imaging, the architecture has the capability to achieve good compression ratio while maintaining critical visual

details. The ability to handle large datasets with very minimal loss of quality ensures its usage in hyperspectral imaging applications like remote sensing and medical diagnostics, where visual fidelity and real-time processing are critical. Also, this design is highly scalable and suitable for high performance applications like 4K video processing. The scalability is mainly due to its pipelined and modular architecture that can handle increased data rates by replicating its processing units to accommodate larger datasets. Efficient hardware utilization, reduced area and power consumption also make this design suitable for 4K video processing. The use of high-speed CSLA and Booth multipliers ensures the system maintains low latency and high throughput, even when processing high-resolution video data. These optimizations make the architecture adaptable for future multimedia applications requiring higher computational performance.

The scalability and flexibility of the proposed architecture are also evident in its applications across digital communication systems and multimedia applications. Designed for modern FPGA and CMOS technologies, this architecture provides support for seamless integration into larger systems due to its pipelined structure and optimized resource distribution. Its robustness ensures that it can manage high data volumes without sacrificing performance or image quality. Additionally, the hardware design's compatibility with advanced encoding techniques like Run-Length Encoding (RLE), Huffman coding etc makes it suitable for larger-scale multimedia systems, extending its potential use in streaming services, portable devices, and other high-resolution multimedia applications. This adaptability highlights its capacity to effectively meet the demands of next-generation multimedia technologies.

REFERENCES

[1] M. Siekkinen, E. Masala and J. K. Nurmine, **Optimized upload strategies for live scalable video transmission from mobile devices**, 2017 *IEEE Trans. Mobile Comput*, Vol 16, No 4, pp. 1059–1072, 2017.

[2] Yan Xing, Ziji Zhang, Yiduan Qian, Qiang Li and Yajuan He, **An Energy-Efficient Approximate DCT for Wireless Capsule Endoscopy Application**, *IEEE International Symposium on Circuits and Systems (ISCAS),* 2018.

[3] Hasan Erdem Yantir, Ahmed M Eltawil, Khalid N Salama, **An Efficient 2D Discrete Cosine Transform Processor for Multimedia Applications**, *2020 28th Signal Processing and Communication Applications Conference (SIU), IEEE, 2020.*

[4] Elham Esmaeili, Nabiollah Shiri, Mahmood Rafiee, Ayoub Sadeghi, **A Multiplier-Free Discrete Cosine Transform Architecture Using Approximate Full adder and Subtractor**, *IEEE Embedded Systems Letters, Vol 16, Issue 4, IEEE, 2024.*

[5] M. Masera, M. Martina and G. Masera, **Adaptive approximated DCT architectures for HEVC**, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 27, No 12, pp. 2714–2725, 2017.
[6] Ahmet Can Mert, Ercan Kalali and Ilker Hamzaoglu, **High Performance 2D Transform Hardware for Future Video Coding**, *IEEE Transactions on Consumer Electronics*, 2017.
[7] M. Chen, Y. Zhang and C. Lu, **Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms**, *AEU-Int. J. Electron. Commun*, Vol 73, pp. 1–8, 2017.
[8] S. Chatterjee and K. Sarawadekar, **An optimized architecture of HEVC core transform using real valued DCT coefficients**, *IEEE Transactions on Circuits and Systems II, Express Briefs*, Vol 65, No 12, pp. 2052–2056, 2018.
[9] Moustafa Masoumi, HamidReza Ahmadifar, **Performance of HEVC discrete cosine and sine transforms on GPU using CUDA,** *IEEE 4th International conference on Knowledge-Based Engineering and Innovation (KBEI), IEEE, 2017.*
[10] Ashish Singhadia, Meghan Mamillapalli and Indrajit Chakrabarti, **Hardware-Efficient 2D DCT/IDCT Architecture for Portable HEVC-Compliant Devices**, *IEEE Transactions on Consumer Electronics*, 2020.
[11] Ahmad Shabani, Mohammad Sabri, Bahareh Khabbazan and Somayeh Timarchi, **Area and Power Efficient Variable-Sized DCT Architecture for HEVC Using Muxed-MCM Problem**, *IEEE Transactions On Circuits And Systems—I:Regular Papers*, Vol 68, No 3, 2021.
[12] Radhika R, Dinesh S, Harini S, Kaviya S, Mallaadi Mohammad Hammed, Priyadarshini M, **Power and Area Optimization Techniques for Reconfigurable Inverse Discrete Cosine Transform FPGA for High Performance Computation Electromagnetics,** *9th International Conference on Smart Structures and Systems (ICSSS), IEEE, 2023.*
[13] D. Mukherjee and S. Mukhopadhyay, **Hardware efficient architecture for 2D DCT and IDCT using Taylor-series expansion of trigonometric functions**, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 30, No 8, 2020.
[14] A. C. Mert, E. Kalali and I. Hamzaoglu, **An FPGA implementation of future video coding 2D transform**, *Proc. IEEE 7th Int. Conf. Consum. Electron. -Berlin (ICCE-Berlin)*, pp. 31–36, 2017.
[15] W. Imen, B. Fatma, M. Amna and N. Masmoudi, **DCT -II transform hardware-based acceleration for VVC standard**, *Proc. IEEE Int. Conf. Design Test Integr. Micro Nano-Syst. (DTS)*, pp. 1–5, 2021.
[16] C. Loeffler, A. Ligtenberg and G. S. Moschytz, **Practical fast 1-D DCT algorithms with 11 multiplications**, *Proc. Int. Conf. Acoust., Speech, Signal Process, IEEE*, Vol 2, pp. 988–991, 1989.

[17] Zhiwei Zhou and Zhongliang Pan, **Effective Hardware Accelerator for 2D DCT/IDCT Using Improved Loeffler Architectur,** *IEEE Access,* 2022.

[18] Aiswarya Simson and Deepak S, **Design and Implementation of High-Speed Hybrid Carry Select Adder**, *2021 International Conference On Advances In Electrical, Computing, Communication and Sustainable Technologies (ICAECT), IEEE*, 2021.

[19] Jasmine Saini, Somya Agarwal and Aditi Kansal, **Performance, Analysis and Comparison of Digital Adders**, *International Conference on Advances in Computer Engineering and Applications (ICACEA), IEEE*, 2015.

[20] A. N. M. Hossain and M. A. Abedin, **Implementation of an XOR Based 16-bit Carry Select Adder for Area, Delay and Power Minimization**, *International Conference on Electrical, Computer and Communication Engineering (ECCE), IEEE*, 2019.

[21] Milad Bahadori, Mehdi Kamal, Ali Afzali-Kushaa and Massoud Pedramb, **An energy and area efficient yet high-speed square root carry select adder structure**, *Computer and Electrical Engineering, Elsvier*, 2017.

[22] B. Ramkumar and Harish M Kittur, **Low-Power and Area-Efficient Carry Select Adder**, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol 20, No 2, 2016.

[23] Nidhi Gaur, Anu Mehra and Pradeep Kumar, **16 Bit Power Efficient Carry Select Adder**, *International Conference on Signal Processing and Integrated Networks (SPIN), IEEE*, 2019.

[24] Deepthi Obul Reddy and P. Ramesh Yadav, **Carry Select Adder with Low Power and Area Efficiency**, *International Journal of Research and Development*, Vol 3, No 3, pp. 2935, 2012.

[25] G. Kishore Kumar and N. Balagi, **Reconfigurable Delay Optimized Carry Select Adder**, *IEEE International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICIEEIMT)*, 2017.

[26] S. Kalaivani and T. Jayachandran, **Low Power Square Root Carry Select Adder,** *Journal of Network Communications and Emerging Technologies (JNCET)*, Vol *5, No* 1, 2015.

[27] Nikhil Patil, Ashish Joshi, Nima Eskandari and Tooraj Nikoubin, **RCA with Conditional BEC in CSLA structure For Area-Power Efficiency**, *6th ICCCNT, IEEE*, 2015.

[28] Siva. S. Sinthura, Afreen Begum, B. Amala, A. Vimala and V. Vidhya Aparna, **Implemenation and Analysis of different 32-bit multipliers on aspects of Power, Speed and Area**, *Proceedings of the 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE*, 2018.

[29] P. Kavipriya, S. Lakshmi, T. Vino, M.R. Ebenezar Jebarani and G. Jegan, **Booth Multiplier Design Using Modified Square Root Carry-Select-**

Adder, *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), IEEE*, 2021.

[30] Pragati Dahiya, Priyanka Jain, **Efficient Recursive Algorithm for Discrete Cosine Transform and Inverse Discrete Cosine Transform**, *International Conference on Sustainable Energy, Electronics and Computing Systems, IEEE, 2018.*