PAPER • OPEN ACCESS

Implementation of Two-Dimensional (2D) Discrete Cosine Transform (DCT) using Reversible Gates

To cite this article: Sathesh Aravindh and S M Sakthivel 2020 *J. Phys.: Conf. Ser.* **1716** 012036

Implementation of Two-Dimensional (2D) Discrete Cosine Transform (DCT) using Reversible Gates

Sathesh Aravindh[1], **S M Sakthivel**[2]

[1,2] School of Electronics Engineering, Vellore Institute of Technology, Chennai, India

Email:[1]aravindhsah1598@gmail.com, [2]sakthivel.sm@vit.ac.in

Abstract. This research work presents the application-specific integrated circuit (ASIC) implementation of one dimensional (1 × 8) and two dimensional (8 × 8) discrete cosine transform using the reversible logic gates. During the DCT implementation, the integer DCT method of realization is adopted, thereby eliminating the occurrence of floating-point operations in the data path architecture. Thus, the integer mode of DCT realization using reversible logic gates in the proposed DCT architecture improves the delay, power and area of the 1D and 2D DCT data path structure when compared with the normal real DCT implementation. The proposed methodology of DCT implementation includes the construction of full adder and full subtractor using reversible gates and then extended for the realization of the DCT butterfly structure. Especially in this research method, Peres reversible gate is used for the implementation of full-adder and Thapliyal Ranganathan (TR) gate for the implementation of full subtractor, respectively. Thus, the 1D and 2D DCT implementation using reversible logic gates in 180nm technology nodes consume an overall area of 55504.311 sq.µm and 1122324.044 sq.µm respectively.

1. Introduction

Discrete cosine transform is one of the widely used image transforms for most of the image processing applications such as image compression, watermarking and noise removal. The primary reason for using the DCT rather than other transforms such as DWT, SVD and Hadamard is due to its high energy compaction of frequency coefficients and low computational complexity. Generally, in the discrete cosine transform, the signal is analyzed by individual frequency components such as low, mid and high-frequency components [1]. The general equation for a 1D (*N* input terms) and 2D DCT are given by equations 1 and 2.

$$\boldsymbol{F(u) = (2/N)^2 \sum_{i=0}^{N-1} \left(\mathrm{A(i)}.\cos\frac{\pi.u}{2N}(2i+)\right) f(i)} \tag{1}$$

$$\boldsymbol{F(u,v) = (2/N)^2(2/M)^2 \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(\mathrm{A(i)}.\mathrm{A(j)}.\cos\frac{\pi.u}{2N}(2i+1)\cos\frac{\pi.u}{2M}(2j+1)\right) f(i,j)} \tag{2}$$

In the above equation f(i), f(i,j) represents the input image signals and F(u), F(u,v) represents the output DCT transformed signals. Also, the literals i, j, M and N indicate the row and column size of the respective input and output signals. Moreover, the majority of the DCT realizations are carried out using real DCT form which involves floating-point operations, thereby making the implemented architecture more complex with the increased area, power and delay. Therefore, in the present work to overcome this limitation, the implementation of the 1D-DCT and 2D-DCT is carried out with the help of the Integer DCT algorithm, which does the operations using only shift and add operations [2] which eventually eliminates the floating-point operations in the DCT transform. Thus, the integer DCT implementation using reversible logic gates structures is implemented using this logic [3, 4] results in high performance and reduced delay hardware architecture. Also, most of the DCT implementations are carried out using software-based realization, which lacks performance improvement in terms of area, power and delay. Henceforth in the present work, the ASIC implementation of DCT improves the performance [5] when compared with software implementation and also due to the presence of a dedicated circuit, the proposed DCT data path can be incorporated as hardware accelerator during SOC implementation [6]. The rest of the paper is organized as follows: section -2 describes the basic concepts of reversible logic followed by the working principle of basic reversible logic gates and its applications. Then in section 3, the proposed integer DCT algorithm with the corresponding butterfly structure is elaborated. Following the section 3, the proposed VLSI architecture implementation is briefly explained in section 4, and finally, in the section 5, the results and discussion of the proposed DCT data path with the corresponding output and inferences are explained.

2. Basic concept of Reversible Logic

In this section, the basic operation of reversible logic and its need for the present architectural implementation is explained. Then the working principle of basic reversible gates with the equivalent structure are also elaborated.

2.1 Reversible Logic Concept

Reversibility in computing prevents loss of information about any computational state in the processing of data such that any data from the previous states, if required, can be obtained by backtracing the algorithm or undoing the process [7]. This is termed as logical reversibility. For example, an inverter (logical NOT gate) is considered a reversible gate because the input can be obtained by back tracing the outputs. The exclusive or (XOR) gate is not considered to be a reversible one due to the fact that its two inputs cannot be brought back in any way, from its output. However, there exists a controlled-NOT gate (CNOT), which happens to be a XOR gate with reversibility achieved, can be defined by saving one of the inputs and can implement any reversible Boolean function. The pros of logical reversibility are obtained only after achieving physical reversibility. Perfect physical reversibility can’t be achieved practically. Computing Networks radiates heat when there is a change in voltage level and there is a change in the value of input/output bits. Heat dissipation accounts for the major component of the energy utilized. Instead of oscillating between various voltage levels, reversible logic elements will relocate charge to another point gradually. This way, the amount of energy lost per each switching is reduced. Reversible logic elements play a major in recovering the input states from the outputs. Eventually, for providing optimal efficiency, they also must imbibe the property of reversibility.

2.2 Reversible Logic Gates

Reversible logic gates are devices where the number of inputs is equal to the number of outputs [8]. In this kind of devices, there is a privilege of obtaining back the inputs. In this synthesis of reversible circuits, concepts violating reversibility like direct fan-out are not allowed. However, fan-out can be achieved by using adding various other gates. A circuit with reversibility is obtained by using optimum number of gates with reversibility. For designing a reversible circuit, there are a few factors that are to be considered for attaining the performance intended by the circuit.

- The number of gates that are used in the circuit for obtaining the functionality.
- Of all the inputs given, the number of them that are required to be maintained at logic one or zero to ensure the functionality.
- Garbage outputs obtained at the output port of each reversible logic gates. These are crucial for achieving reversibility.

- ➢ The entire circuit's cost is calculated in terms of cost of an individual gate. Greater the number of individual gates in the circuit greater the cost.

2.3 Basic Reversible Logic Gates

In this section, the two basic gates Peres and TR gates operation is elaborated with its internal structure. During the implementation using reversible gates, there exist three primary considerations such as: i) Quantum Cost - for minimizing the complexity, ii) Delay – how effectively the gate propagates the input and makes output and iii) Garbage value - the number of inputs and outputs not used in the implementations. By considering the aforesaid valid reasons, we have taken Peres and TR reversible gate for our internal architecture implementation.

*2.3.1 Peres gate.*The Figure 1 shows a 3-input 3-output Peres gate. Wherein A, B, C are the inputs and P, Q and R are the outputs [8]. The output is interpreted by P = A, Q = A^ B and R=(A&B) ^ C. Peres gate is used in the proposed design due to its relatively lower quantum cost.

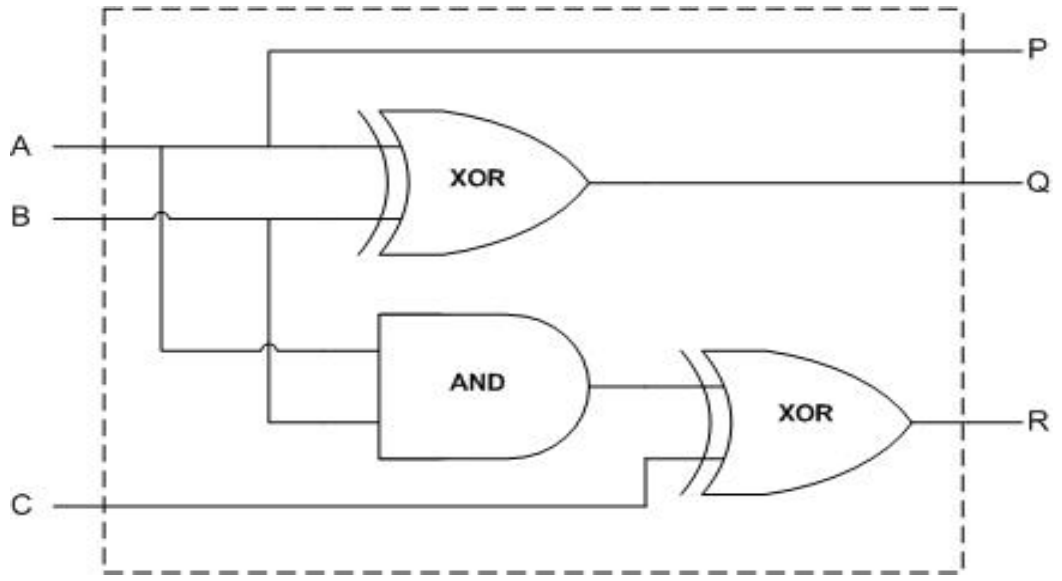

Figure 1. Peres Reversible gate internal structure.

In the proposed methodology, the same Peres gate is used to realize the full adder structure as the internal DCT transform architecture involves only addition and subtraction only. Therefore, for realizing the addition operation, the adder circuit is implemented using the Peres gate. In this implementation, first, the full adder circuit is implemented and using the same full adder the 8-bit adder circuit is implemented. The constructed full-adder logic using the Peres reversible gate is shown in Figure 2.

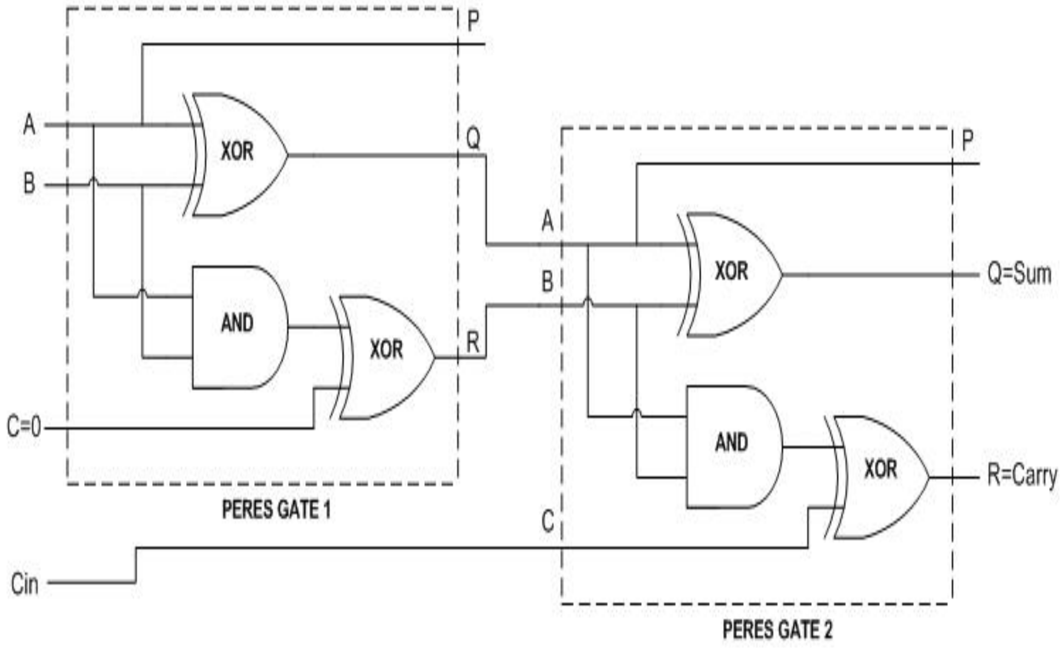

Figure 2. Reversible Full Adder circuit using two Peres gates.

*2.3.2 Thapliyal Ranganathan gate (TR gate).*Similarly, for the implementation of subtraction operation, the Thapliyal Ranganathan gate shown in Figure 3 is incorporated. The TR gate consists of 3-input 3-output, Where A, B, C are the inputs and P, Q, R are the outputs. The output is interpreted by P =A, Q =A^B, R = (A & (~B)) ^C. Thapliyal and Ranganathan proposed Reversible Binary Subtractor using TR Gate [9]. The functionality similar to binary subtraction is obtained using TR gate. The minimum number of reversible gates, Garbage outputs are used for the effectiveness of the circuit and it is ensured that the Quantum Cost also stands low [9].

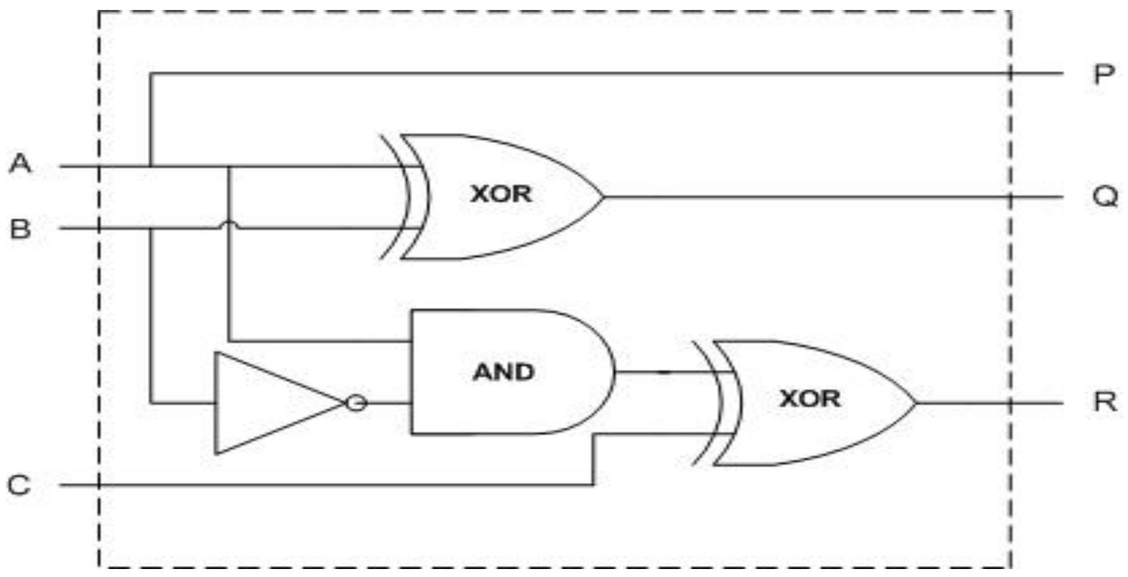

Figure 3. Thapliyal Ranganathan Reversible Gate.

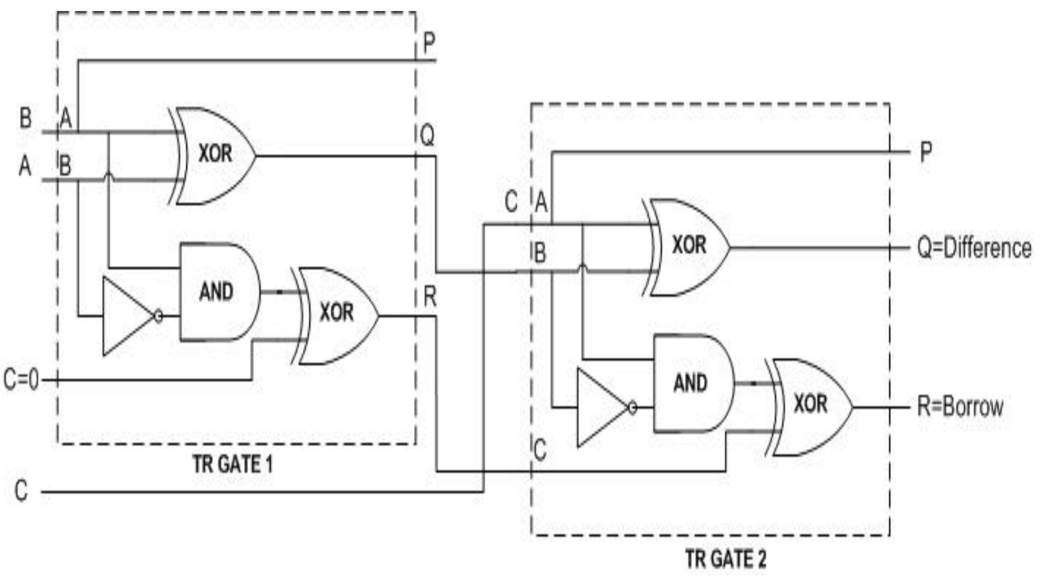

Figure 4. Reversible Full Subtractor using two TR gates.

After implementing the TR gate, the full subtractor in the Figure 4 is realized using the constructed TR gate for single bit subtraction. Subsequently, the same full subtractor configuration is iteratively used in the proposed methodology to construct the 8-bit subtractor for carrying out the subtraction in the integer DCT butterfly-based transform process.

2.4. Applications of Reversible Gates
Reversibility computation architecture has found its application in the areas of pc security, fast processing (low latency) architectures and cryptography. However, the most long-run profit is felt at places where high energy potency, speed and performance is needed.
Some of the reversible logic gates-based applications are
1. Quantum Computing
2. Low power circuits
3. Nanotechnology
4. Signal processing
5. Optical Computing

Additionally, the field-programmable gate arrays (FPGAs) based reversible realization are also used in CMOS technology for low power, high testability and optimum delay [10].

3. Algorithm for Integer DCT (1D)
In the 1D integer DCT operation, the transformation process is orthogonally realized using the addition and subtraction operations. For example, in the 8 point 1-D DCT, 8 different inputs are introduced to the structure parallelly and these inputs are passed through a series of stages which include add and shift operations. The output of each stage is stored and then processed to the next stage to ensure that the values can be restored by backtracking the algorithm.

All these stages in the algorithm include basic arithmetic operations and shift and add of few operators is done where ever necessary. Eight different inputs are put into the initial stage of the structure and after processing by subsequent stages in reference to the clock pulses, the output is obtained from the final stage. The entire operation of the 1D transformation for each of the stage is given below with the set of addition and subtraction operations. In the DCT calculations, the a0 to a7 indicates the inputs, d0 to d7 represents the outputs where the remaining literals indicate the intermediated stored buffer values. Here each input coefficients, intermediate pixel values and output coefficients are operated on an eight-bit precision format to achieve the output accurately. Also, in the calculation, the constants K1, K2, K3 and K4 are selected based on the butterfly structure of the DCT transformation. Also, the entire operation of 1D forward integer DCT operation is pictorially represented in Figure 5 for 8 input and outputs (i.e., for 8-point 1D-DCT).

The step by step procedure of the algorithm is stated as below for each stage with the respective equations:

Stage – a:

$a0 = i0 + i7 \quad a1 = i1 + i6 \quad a2 = i2 + i5 \quad a3 = i3 + i4$

$a4 = i0 - i7 \quad a5 = i1 - i6 \quad a6 = i2 - i5 \quad a7 = i3 - i4$

Stage – b:

$bo = a0 + a3 \quad b1 = a1 + a2 \quad b2 = a0 - a3 \quad b3 = a1 - a2$

$$b4 = \frac{a7}{4} + a4 + \frac{a4}{4} - \frac{a4}{16}$$

$$b5 = a5 + a6 - \frac{a6}{4} - \frac{a6}{16}$$

$$b6 = a6 - a5 + \frac{a5}{4} + \frac{a5}{16}$$

$$b7 = \frac{a4}{4} - a7 - \frac{a7}{4} + \frac{a7}{16}$$

Stage – c:

$c0 = b0 + b1 \quad c1 = b0 - b1 \quad c2 = b2 + \frac{b2}{4} + \frac{b3}{2} \quad c3 = \frac{b2}{2} - b3 - \frac{b3}{4}$

$c4 = b4 + b5 \quad c5 = b4 - b5 \quad c6 = b6 + b7 \quad c7 = b6 - b7$

Stage-d:

$d0 = c0 \quad d1 = c1 \quad d2 = c2 \quad d3 = c3$

$d4 = c4 \quad d5 = c5 + c7 \quad d6 = c5 - c7 \quad d7 = c7$

Where each variable carries 8-bit data (i.e., $i0[7{:}0]$). Similarly, the K values are given as

$K1 = \frac{1}{4}, K2 = \frac{11}{16}, \; K3 = \frac{19}{16}, \; K4 = \frac{1}{2} \; and \; K5 = \frac{3}{4}$

In these 1D integer DCT operation, the DCT coefficients are calculated only using the integer arithmetic's such as addition, subtraction and shifting only. Thus, the implemented VLSI data path architecture using the integer DCT transform has low computational intense architecture thereby avoiding the floating-point operations.

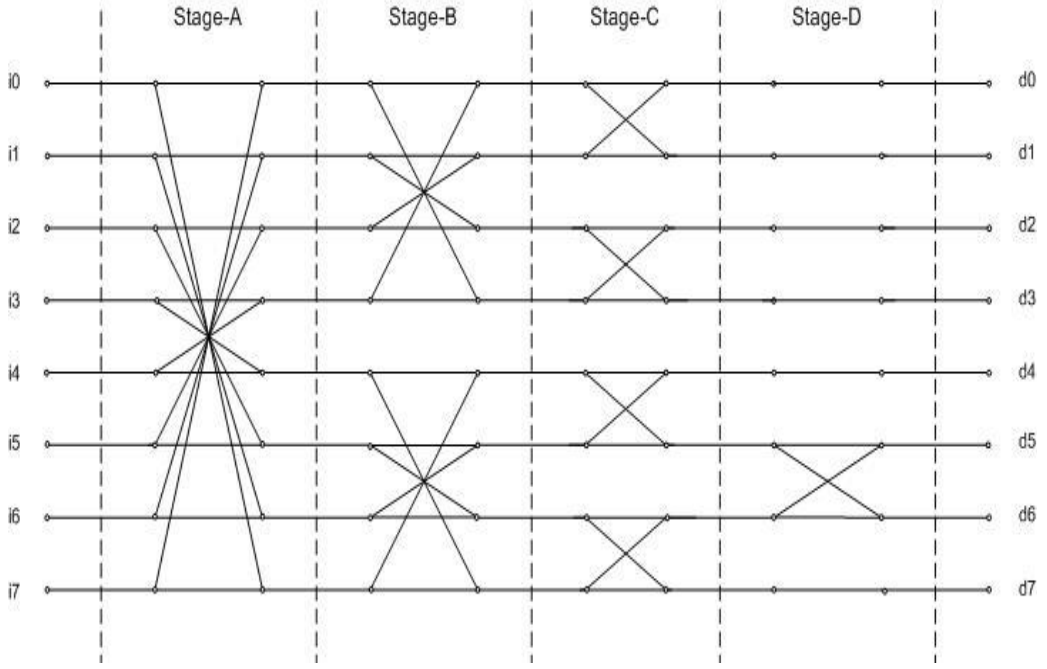

Figure 5. Signal flow graph for 1D-DCT in forward mode.

4. Proposed 2D integer DCT data path architecture

In the proposed implementation of 2D Integer DCT algorithm, the 1D DCT transformation explained in section 3 is incorporated row-wise and column-wise. For the 1D DCT operation, the eight-input data are fed for each clock cycle and the corresponding output for input image pixel values is generated. This process of feeding inputs to the structure happens in 8 clock cycles, a single input to all the 8 1DCT's at a clock edge and then the next inputs at the next clock edge and this happens for 8 clock edges. Thus, for implementing an 8 × 8 2D DCT architecture, the 1D DCT data path is implemented eight times in a parallel manner to perform a row-wise DCT operation. In other words, using the eight parallel 1D DCT structure (i.e. 8 × 1) the row-wise DCT operation is carried out. After processing the 1D DCT, the output coefficients are stored in transpose memory and the same eight 1D DCT parallel structure is used to perform the column-wise DCT operation. That is, DCT transform is carried out on the 8 × 1 signal elements to get 8 × 8 DCT transformed output. During the DCT implementation, once all the inputs are given, the input to the memory is blocked. That is the 1-D DCT stores the inputs directly into the memory for a particular number of clock pulses depending on the stage at which the 1D-DCT structure is used in the architecture of 2D-DCT algorithm. This is due to the fact that the second stage of 1-d DCT needs to be in idle state for a fixed amount of time i.e., until the transpose stage of the 2D-DCT architecture produces outputs.

Based on the aforementioned 2D integer DCT operation, the eight 1D DCT architecture is iteratively used two times row-wise and column-wise with a transpose memory as shown in the Figure 6. Therefore, for implementing the 8 × 8 DCT transformation is operation is implemented by the architecture as shown in Figure 6. In the Figure 6, the first eight 1D DCT unit performs the row-wise DCT transformation and stores the output in the transpose memory. Then the second eight parallel 1D DCT unit performs the column-wise integer DCT operation on the values of the transpose memory to get the 8 × 8 DCT output. In the proposed architecture, the DCT operation is carried out using the integer DCT butterfly mode-based realization, which involves addition, subtraction and sifting rather than floating-point operation. Therefore, the proposed architecture consumes comparatively less area and power, thereby making the architecture more efficient in performance.

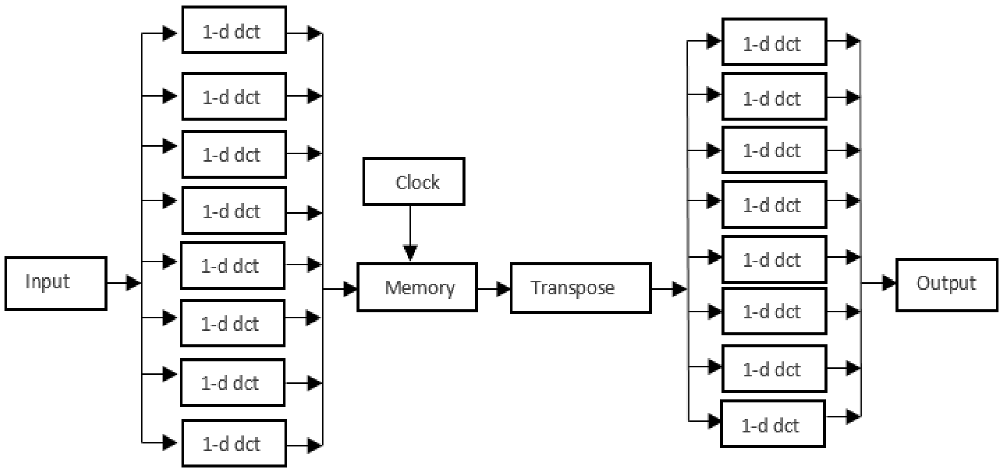

Figure 6. Proposed 2D DCT data path using two 1D DCT data path structure.

The internal butterfly structure of the 1D integer DCT transform is shown in the Figure 7. It consists of eight inputs and 8 outputs and the entire operation is processed in four stages using the corresponding equations given in section-3. The 8-point 1D-DCT has been implemented with the help of the Integer DCT algorithm. Here the literals I indicate the inputs and d represents the outputs and K_i are scaling factors. In the above DCT, there are blocks named from 'a' to 'd', which are named after 8-bit full adders and subtractors, which are implemented using Peres gate Full adder & Thapliyal Ranganathan Gate Full Subtractor. This 1D-DCT architecture requires only 10 multipliers and 26 adder & 13 subtractors [1] each. The product obtained by multiplying with the input DCT coefficients is stored in Read-only memories, so the result can be obtained by comparing it with the Look-Up Table (LUT) multipliers. It results in a highly regular circuit and consists of only basic logic elements.

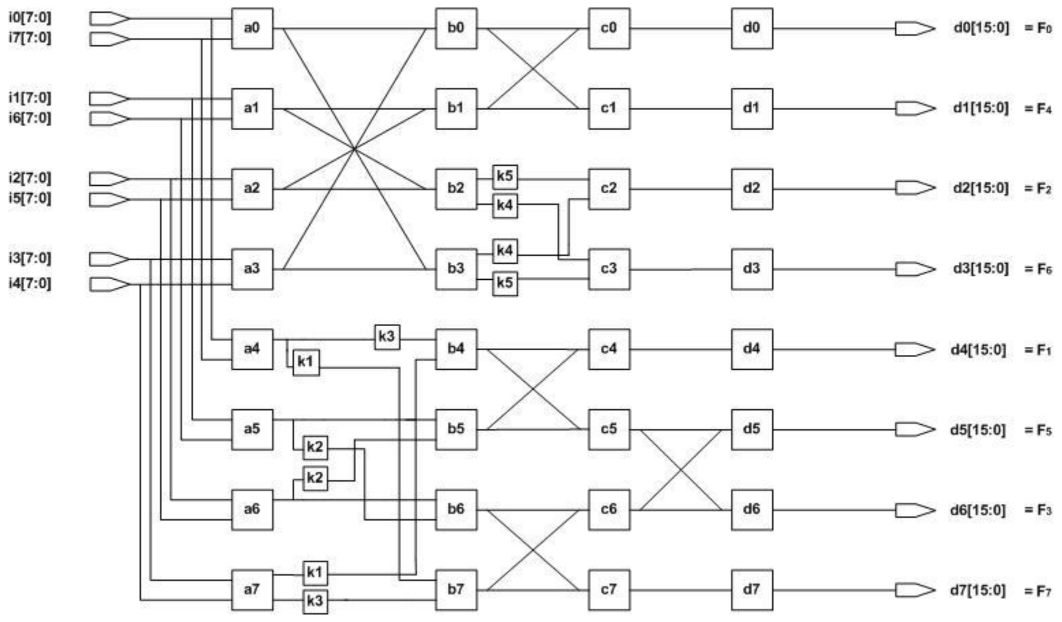

Figure 7 1D-DCT butterfly internal data path structure.

5. Results and Discussion

In the present work, initially, the reversible gates Peres and TR are modelled using Verilog HDL and then using the gates the full adder and full subtractor are constructed respectively. Subsequently, the eight-bit adder and eight-bit subtractor are constructed using the reversible full adder and full subtractor. Then using the constructed full adder and full subtractor, the 1D DCT data path architecture for 8 × 1 transform is modelled using structural modelling. Finally, with the incorporation of transpose memory structure, the 2D DCT architecture is constructed by instantiating the 1D DCT data path two times, as shown in the architecture in Figure-6. After modelling the 2D DCT module, its functionality is verified using logical simulation with appropriate 8 × 8-pixel data input values. The simulation output response of the 2D DCT transform in normal integer realization and reversible logic-based realization are given in the Figure-8 and 9, respectively. During the simulation, the input values are initialized as 8-bit binary data input and integer 2D DCT transform is performed on the 8-bit pixel input data using addition, subtraction and shifting operations only. The functional simulation is carried out using the NCLAUNCH Cadence® simulator.

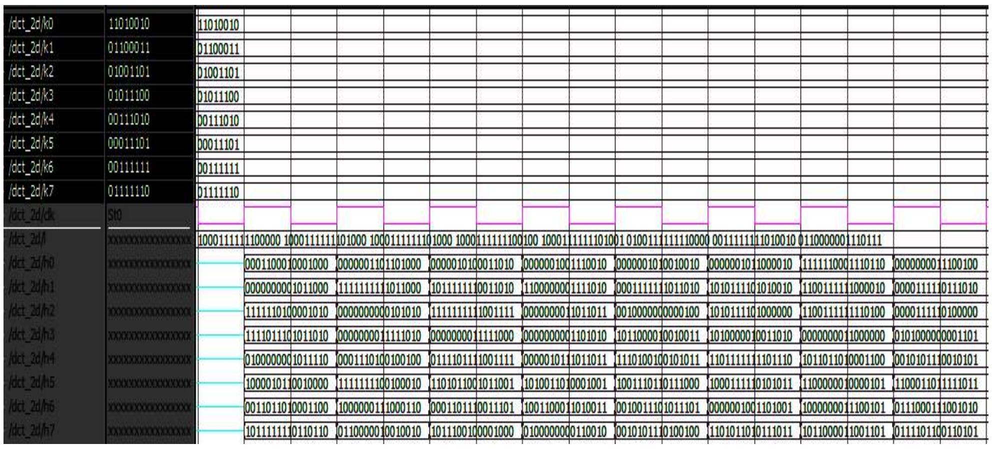

Figure 8. Functional simulation response of integer 2D DCT.

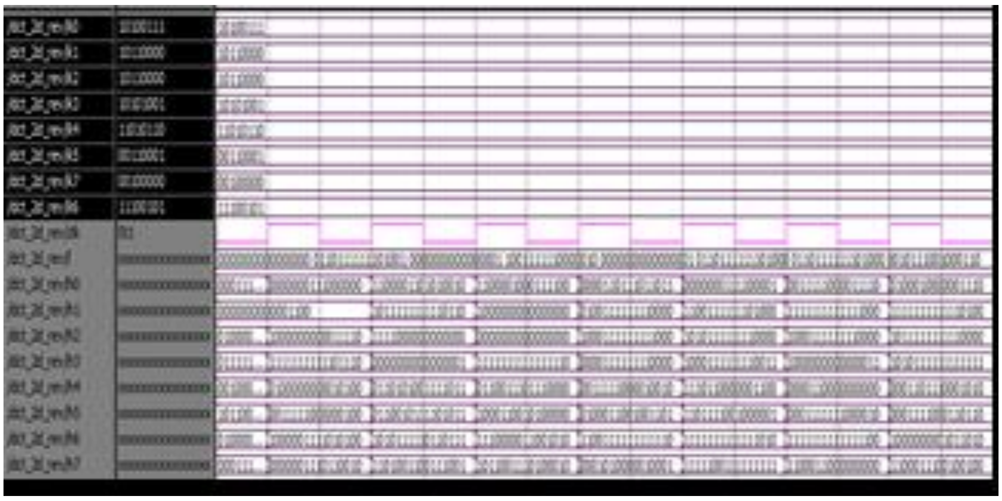

Figure 9. Functional simulation response of Reversible integer 2D DCT.

After simulating the 2D DCT Verilog codes, the respective data path modules are synthesized using GENUS Cadence® RTL compiler in 180 nm technology. The synthesized netlist of the 8-bit full adder and full subtractor are shown in Figure 10 and 11, respectively. After the synthesis, the performance evaluation of the implemented data path is carried out using the parameters Area, Power, Delay and Cell count. The respective values of the parameters are given in Table-1 and also plotted as a bar graph in Figure 11 for visual inference analysis.

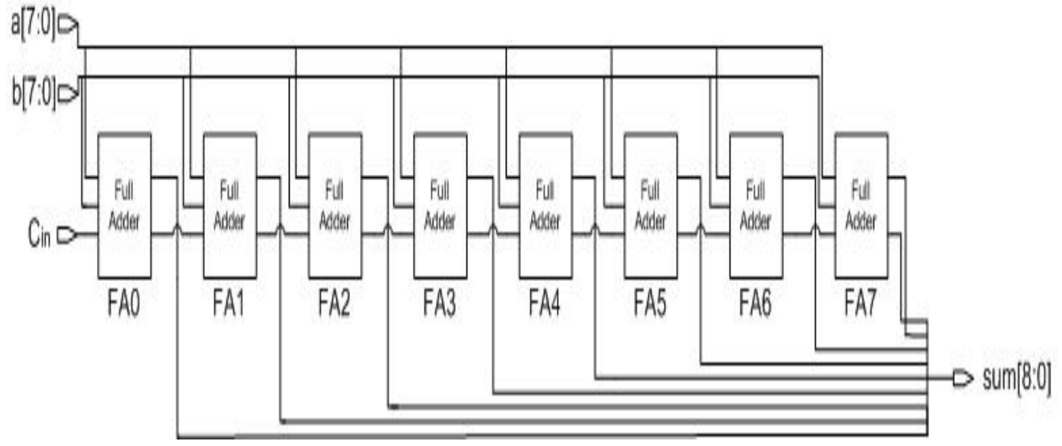

Figure 10. 8-bit full adder synthesized RTL schematic.

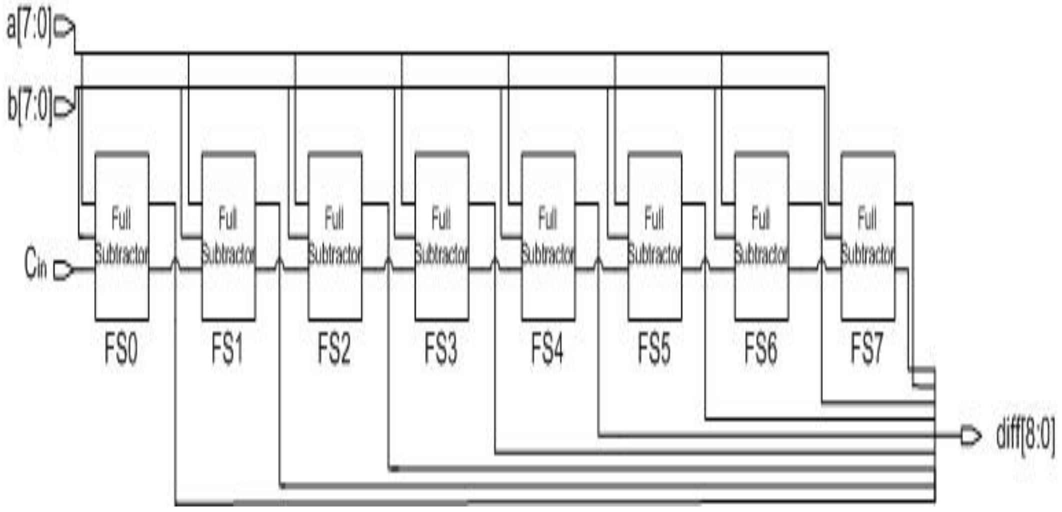

Figure 11. 8-bit Full subtractor synthesized RTL schematic.

Table 1. Performance comparison of DCT architecture in normal and reversible mode

Architecture	Power (nW)		Area (μm²)		Delay (ps)
	Leakage power	Dynamic power	Cell count	Area	
DCT (1D)	2272.723	11644068.52	2078	48918.039	29657
Rev-DCT (1D)	1679.380	17738289.13	2941	55504.311	24563
DCT (2D)	44813.04	122746471.0	41528	1001296.305	29743
Rev-DCT (2D)	33956.32	169427393.8	57336	1122324.044	25487

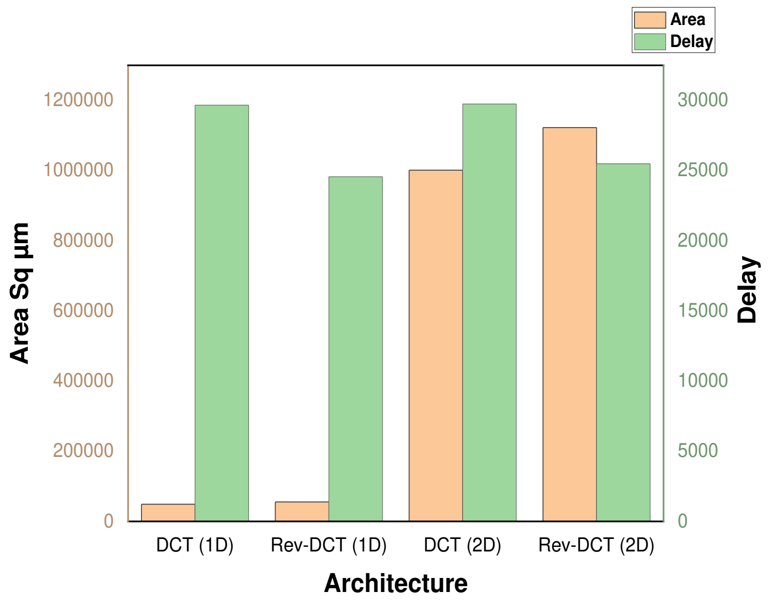

Figure 12. Performance comparison of integer DCT and reversible integer DCT using area and delay parameter.

From the graph shown in Figure 12, it is inferred that the reversible logic-based integer DCT implementation consumes relatively lesser delay when compared with integer DCT implementation with an additional overhead in the area. Thus, from the analysis, we can strongly conclude that the proposed integer DCT 1D and 2D implementation is more suitable for fast processing multimedia devices. Also, the prime reason of that the area and power tend to be on the higher side due to the fact that the reversible gates have n-input and n-output structure which tends to occupy larger area and because there is a large number of garbage outputs in the structure implemented. Also, the dynamic power dissipation tends to be high. In contrast, the leakage power is low in Reversible 2D-DCT. The delay for the Reversible 2D-DCT is expected to be lower on the comparison.

6. Conclusion

In this research work, an ASIC based implementation one dimensional (1 × 8) and two dimensional (8 × 8) discrete cosine transform using the reversible logic gates is carries out using Cadence® EDA tool in 180 nm technology. Also, in the implementation of the DCT data path, the Peres reversible gate-based adder and TR reversible gate based subtractor are designed for performing the addition and subtraction in the DCT butterfly structure. Thus, by realizing the DCT in integer mode, the proposed method uses a minimum number of multipliers, adders & subtractors, which eventually reduces the circuit complexity and delay when compared with the real DCT based architectures. Further, in the present work the proposed integer DCT is functionally verified using logical simulation by NCLAUNCH tools and synthesized using GENUS tool. The implemented 1D and 2D reversible DCT consumes an area of 55504.311 sq.µm and 1122324.044 sq.µm, respectively. Additionally, in the reversible integer DCT based realization, the delay is comparatively reduced on comparing with normal integer DCT implementation.

References

[1] Dahiya P and Jain P 2020 Efficient MDCT recursive structure for VLSI implementation *Circuits Systems and Signal processing* **1** 1-15.
[2] Sakthivel S M and Ravi Sankar A 2020 Compuation efficient image wateramrking architecture with improved performance. Computers and Electrical Engineering 84(**1**) 106649- 106666.
[3] Nibouche, Omar & Boussakta, Said & Darnell, Michael & Benaissa and Mohammed 2010

Algorithms and pipeline architectures for 2D FFT and FFT-like transforms *Digital Signal Processing* **20** 1072-1086. 10.1016/j.dsp.2009.10.028.

[4] Prashan R, Yelekar, Sujata S and Chiwande 2011 Introduction to reversible logic gates & its application *Int. Journal of Computer Applications* **8** 5-9.

[5] Sakthivel S M, Ravi Sankar A 2015 A real time watermarking of grayscale images without altering its content *Proc. of the Int. Conf. on VLSI Systems, Architecture, Technology, and Applications* (Bangalore:India) 8–10

[6] Sakthivel S M, Ravi Sankar A 2016 Real time watermarking of grayscale images using integer DWT transform *Proc. of the Int. Conf. on VLSI Systems, Architecture, Technology, and Applications* (Bangalore:India) 8–10

[7] Kowsalya P and Sakthivel S M 2019 VLSI Implementation of 2D Discrete wavelet transform using reversible logic gates *International Journal of Recent Technology and Engineering (IJRTE)* **8(1)** 1621-1629.

[8] Bhagyalakshmi H R and Venkatesha M K 2010 An improved design of a multiplier using reversible gates *Int. Journal of Engineering Science and Technology* **2(8)** 3838-3845.

[9] Orts Gómez, Francisco José & Ortega Lopez, Gloria & Garzon and E M 2019 A faster half subtractor circuit using reversible quantum gates *Baltic Journal of Modern Computing*.

[10] Vanusha P and Amurtha Vally K 2014 Low power computing logic gates design using reversible logic *Int. Journal of Application or Innovation in Engineering & Management (IJAIEM)* **3(10)**.