iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 5 **Issue:** VI **Month of publication:** June 2017

**DOI:**

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

# Pruning Algorithm for Energy Efficient DCT Implementation for High Efficiency Video Coding

P. Ranjith[1], B. Kedarnath[2], P. Madhusudhan Reddy[3]
[1]Assistant Professor, [2]Professor & Head of the department, [3]Assistant Professor
Electronics and Communication Engineering, Gurunanak Institute of Technology, Hyderabad, India

***Abstract:*** ***This paper presents area and power efficient architectures for the implementation of integer discrete cosine transform (DCT) of different lengths to be used in High Efficiency Video Coding (HEVC).We show that an efficient constant matrix multiplication scheme can be used to derive parallel architectures for 1-D integer DCT of different lengths. We also show that the proposed structure could be reusable for DCT of lengths 4, 8, 16, and 32 with a throughput of 32 DCT coefficients per cycle irrespective of the transform size. Moreover, the proposed architecture could be pruned to reduce the complexity of implementation substantially with only a marginal effect on the coding performance. We propose power-efficient structures for folded and full-parallel implementations of 2-D DCT. From the synthesis result, it is found that the proposed architecture involves nearly 14 percent less area-delay product (ADP) and 19 percent less energy per sample (EPS) compared to the direct implementation of the reference algorithm, on average, for integer DCT of lengths 4, 8, 16, and 32. Also, an additional 19 percent saving in ADP and 20 percent saving in EPS can be achieved by the proposed pruning algorithm with nearly the same throughput rate. The proposed architecture is found to support ultrahigh definition 7680×4320 at 60 frames/s video, which is one of the applications of HEVC.***

***Index Terms—DCT, HEVC, EPS, ADP, DCT***

## I. INTRODUCTION

Recent years have experienced a significant demand for high dynamic range systems that operate at high resolutions. In particular, high-quality digital video in multimedia devices and video-over-Internet protocol networks are prominent areas where such requirements are evident. Other noticeable fields are geospatial remote sensing, traffic cameras, automatic surveillance, homeland security, automotive industry, and multimedia wireless sensor networks, to name but a few. Often hardware capable of significant throughput is necessary; as well as allowable area-time complexity.

In this context, the discrete cosine transform (DCT) is an essential mathematical tool in both image and video coding . Indeed, the DCT was demonstrated to provide good energy compaction for natural images, which can be described by first-order Markov signals. Moreover,in many situations, the DCT is a very close substitute for the Karhunen-Loève transform (KLT), which has optimal properties. As a result, the two-dimensional (2-D) version of the 8-point DCT was adopted in several imaging standards such as JPEG, MPEG-1, MPEG-2, H.261, H.263, and H.264/AVC .

Additionally, new compression schemes such as the High Efficiency Video Coding (HEVC) employs DCT-like integer transforms operating at various block sizes ranging from 4X4 to 32X32 pixels. The distinctive characteristic of HEVC is its capability of achieving high compression performance at approximately half the bit rate required by H.264/AVC with same image quality. Also HEVC was demonstrated to be especially effective for high-resolution video applications. However, HEVC possesses a significant computational complexity in terms of arithmetic operations. In fact, HEVC can be 2–4 times more computationally demanding when compared to H.264/AVC. Therefore, low complexity DCT-like approximations may benefit future video codecs including emerging HEVC/H.265 systems. Several efficient algorithms were developed and a noticeable literature is available. Although fast algorithms can significantly reduce the computational complexity of computing the DCT, floating-point operations are still required.

Despite their accuracy, floating-point operations are expensive in terms of circuitry complexity and power consumption. Therefore, minimizing the number of floating-point operations is a sought property in a fast algorithm. One way of circumventing this issue is by means of approximate transforms.

The aim of this paper is two-fold. First, we introduce a new DCT approximation that possesses an extremely low arithmetic complexity, requiring only 14 additions. This novel transform was obtained by means of solving a tailored optimization problem aiming at minimizing the transform computational cost. Second, we propose hardware implementations for several 2-D 8-point

approximate DCT. The approximate DCT methods under consideration are (i) the proposed transform; (ii) the 2008 Bouguezel-Ahmad-Swamy (BAS) DCT approximation ; (iii) the parametric transform for image compression; (iv) the Cintra-Bayer (CB) approximate DCT based on the rounding-off function ; (v) the modified CB approximate DCT; and (vi) the DCT approximation proposed in the context of beam forming. All introduced implementations are sought to be fully parallel time-multiplexed 2-D architectures for 8x8 data blocks. Additionally, the proposed designs are based on successive calls of 1-D architectures taking advantage of the separability property of the 2-D DCT kernel.Designs were thoroughly assessed and compared.

## II. OBJECTIVE

In this paper, the proposed design is integer DCT and existing DCT. We will design an proposed design of the folded and full parallel integer 2D DCT and also measure the Area and delay for each design. Based on this analysis we will show the best 2D DCTEase of Use

## III. EXISTING SYSTEM

One key feature of HEVC is that it supports DCT of different sizes such as 4, 8, 16, and 32. Therefore, the hardware architecture should be flexible enough for the computation of DCT of any of these lengths. The existing designs for conventional DCT based on constant matrix multiplication (CMM) and MCM can provide optimal solutions for the computation of any of these lengths, but they are not reusable for any length to support the same throughput processing of DCT of different transform lengths. Considering this issue, we have analyzed the possible implementations of integer DCT for HEVC in the context of resource requirement and reusability, and based on that, we have derived the proposed algorithm for hardware implementation. We have designed scalable and reusable architectures for 1-D and 2-D integer DCTs for HEVC that could be reused for any of the prescribed lengths with the same throughput of processing irrespective of transform size

## IV. PROPOSED SYSTEM

We propose digital computer architectures that are custom designed for the real-time implementation of the fast algorithms. The proposed architectures employs two parallel realizations of DCT approximation blocks. The first instantiation of the DCT block furnishes a row-wise transform computation of the input image, while the second implementation furnishes a column-wise transformation of the intermediate result. The row- and column-wise transforms can be any of the DCT approximations detailed in the paper.

In other words, there is no restriction for both row- and column-wise transforms to be the same. However, for simplicity, we adopted identical transforms for both steps.

Between the approximate DCT blocks a real-time row-parallel transposition buffer circuit is required. Such block ensures data ordering for converting the row-transformed data from the first DCT approximation circuit to a transposed format as required by the column transform circuit.

### *A. Efficient Four-Point DCT Architecture For HEVC*

*1) General:* We present algorithms for hardware implementation of the hevc integer dcts of different lengths 4, 8, 16, and 32. We illustrate the design of the proposed architecture for the implementation of four-point and eight-point integer dct along with a generalized design of integer dct of length n, which could be used for the dct of length n = 16 and 32. Moreover, we demonstrate the reusability of the proposed system.We propose power-efficient designs of transposition buffers for full-parallel and folded implementations of 2-d integer dct. We propose a bit-pruning scheme for the implementation of integer dct and present the impact of pruning on forward and inverse transforms. We compare the synthesis result of the proposed architecture with those of existing architectures for hevc

*2) Modules*

*a)* Proposed architecture of four-point integer DCT
*b)* Proposed generalized architecture for integer DCT of lengths $N = 8$.
*c)* Proposed reusable architecture for $N = 8$.
*d)* Folded structure of 2-D integer DCT
*e)* Full-parallel structure of 2-D integer DCT.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

*B. MODULE DESCRIPTION*

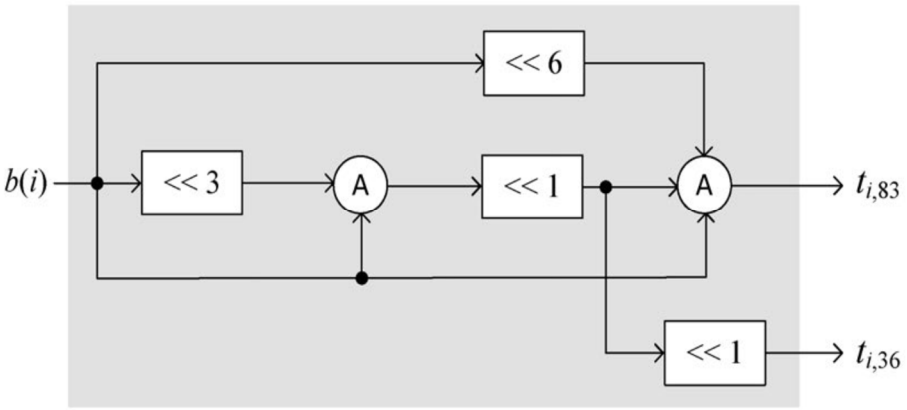
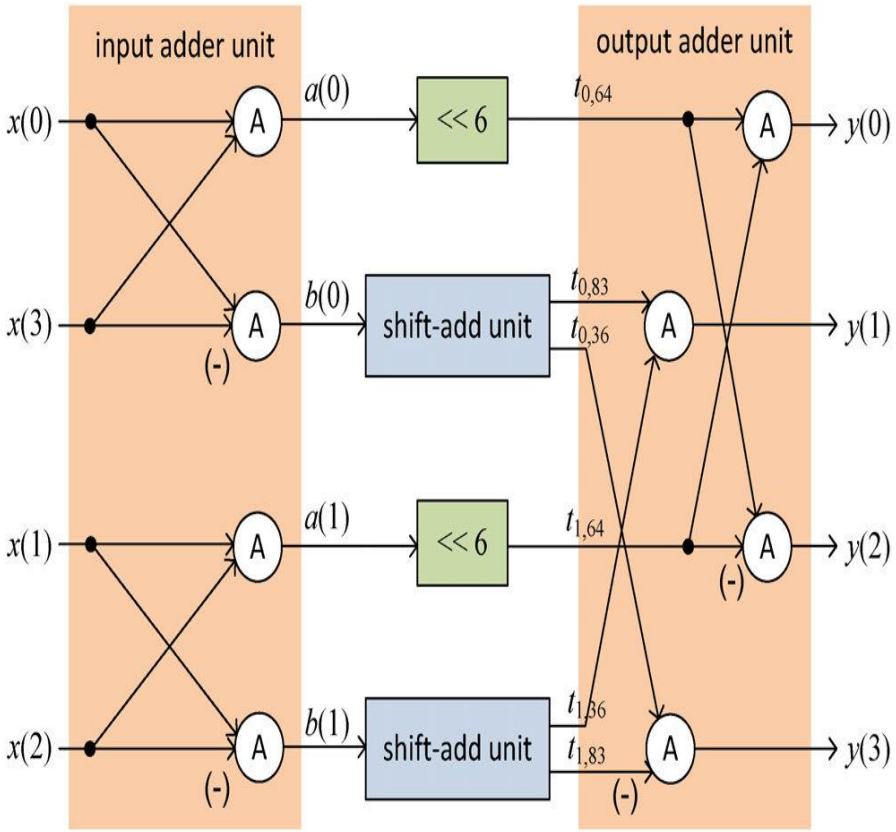



Fig.1 Structure of SAU

*C. PROPOSED ARCHITECTURE OF FOUR-POINT INTEGER DCT*

The proposed architecture for four-point integer DCT is shown in Fig. It consists of an input adder unit (IAU), a shift-add unit (SAU), and an output adder unit (OAU). The IAU computes $a(0)$, $a(1)$, $b(0)$, and $b(1)$ according to STAGE- 1 of the algorithm as described in Table I. The computations of $ti,36$ and $ti,83$ are performed by two SAUs according to STAGE-2 of the algorithm. The computation of $t0,64$ and $t1,64$ does not consume any logic since the shift operations could be rewired in hardware. The structure of SAU is Shown in Fig. Outputs of the SAU are finally added by the OAU according to STAGE-3 of the algorithm.



Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

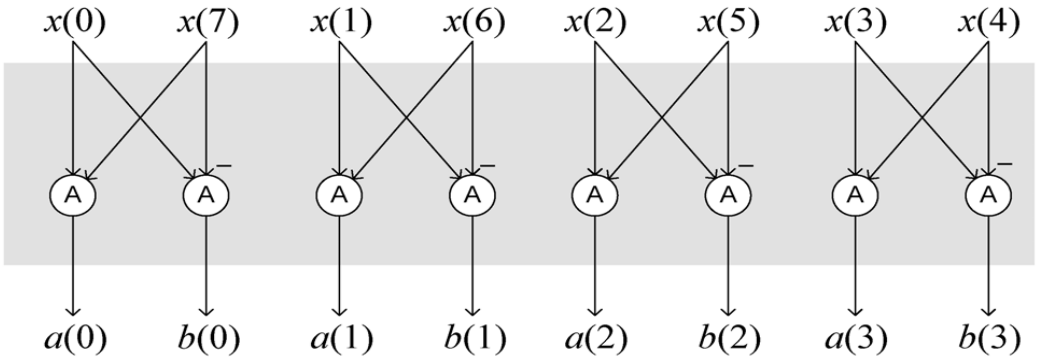



Fig. 2 Structure of IAU

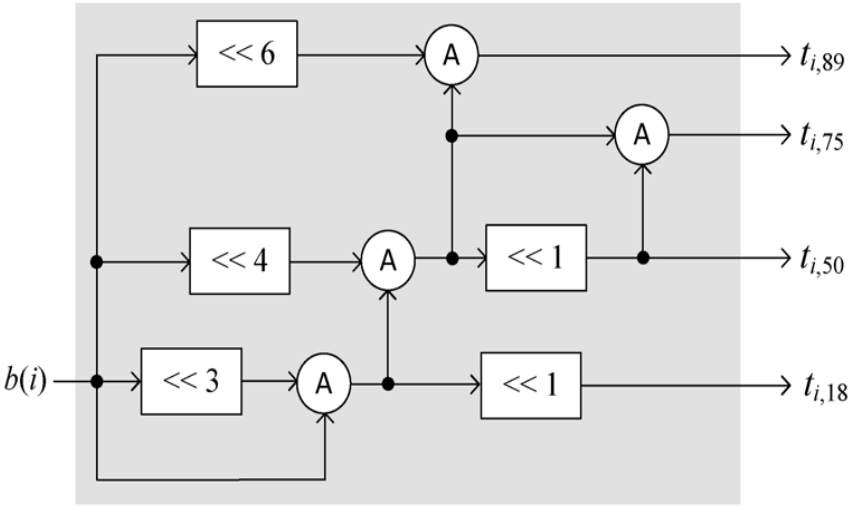



Fig. 3 Structure of SAU

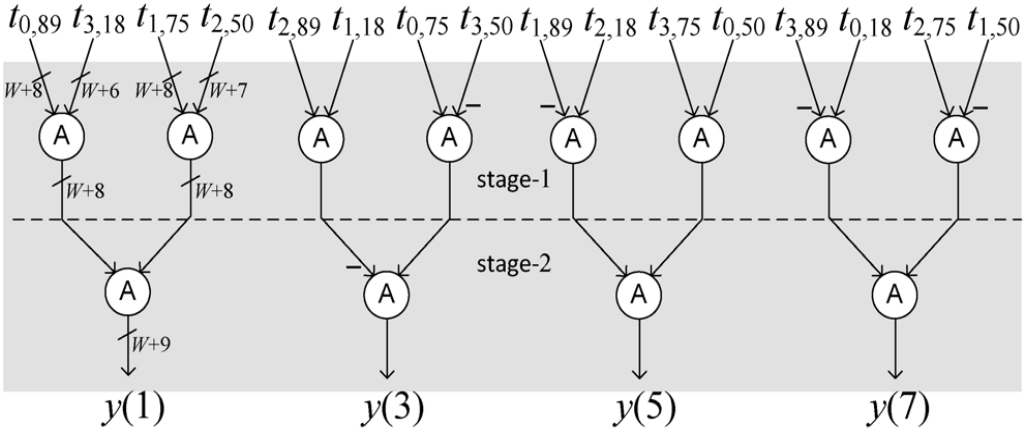



Fig. 4 Structure of OAU

*D. Proposed generalized architecture for integer DCT of lengths N = 8*

The generalized architecture for *N*-point integer DCT based on the proposed algorithm is shown in Fig. It consists of four units, namely the IAU, (*N*/2)-point integer DCT unit, SAU, and OAU. The IAU computes *a*(*i*) and *b*(*i*) for *i* = 0, 1, ...,*N*/2 − 1 according to STAGE-1 of the algorithm of Section II-B. The SAU provides the result of multiplication of input sample with DCT coefficient by

Volume 5 Issue VI, June 2017
ISSN: 2321-9653


STAGE-2 of the algorithm. Finally, the OAU generates the output of DCT from a binary adder tree of log2 $N - 1$ stages, respectively, illustrates the structures of IAU, SAU, and OAU in the case of eight-point integer DCT. Four SAUs are required to compute $ti,89$, $ti,75$, $ti,50$, and $ti,18$ for $i = 0, 1, 2$, and 3 according to STAGE-2 of the algorithm. The outputs of SAUs are finally added by two-stage adder tree according to STAGE- 3 of the algorithm. Structures for 16- and 32-point integer DCT can also be obtained similarly.

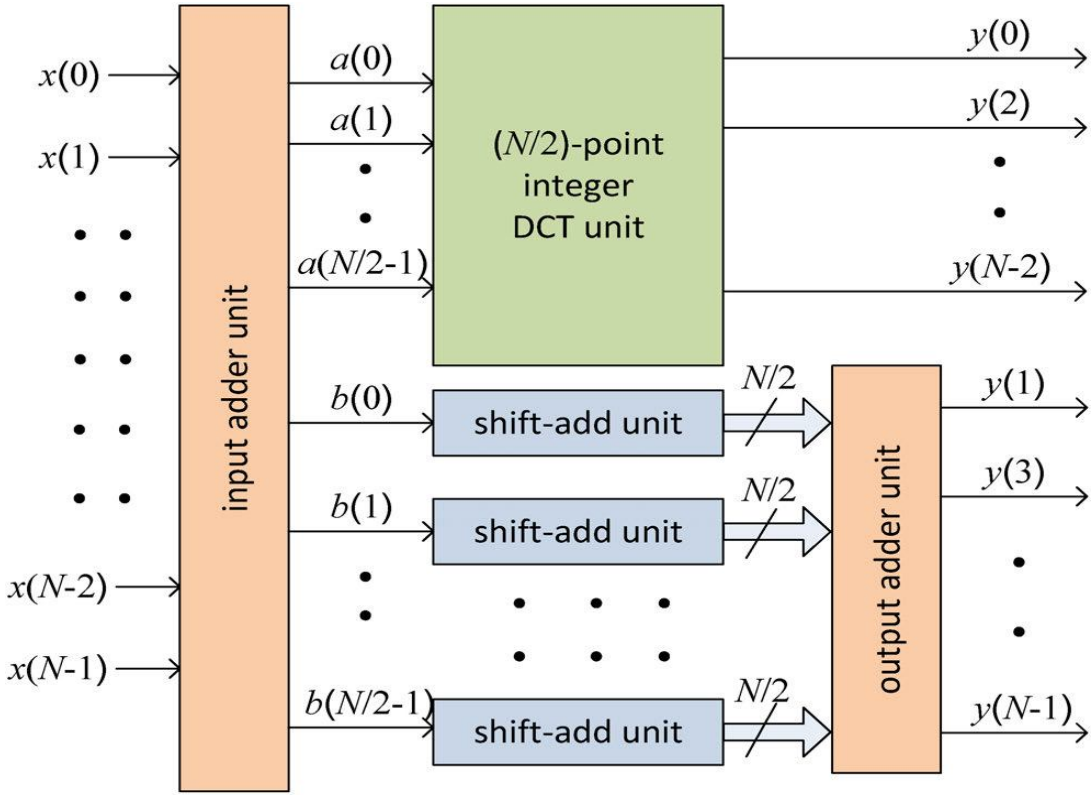



Fig. 5 Proposed generalized architecture for integer DCT of lengths N = 8.

The proposed reusable architecture for the implementation of DCT of any of the prescribed lengths is shown in Fig. 4(a) There are two $(N/2)$-point DCT units in the structure. The input to one $(N/2)$-point DCT unit is fed through $(N/2)$ 2:1 MUXes that selects either $[a(0), ..., a(N/2 - 1)]$ or $[x(0), ..., x(N/2 - 1)]$, depending on whether it is used for $N$-point DCT computation or for the DCT of a lower size.

The other $(N/2)$-point DCT unit takes the input $[x(N/2), ..., x(N - 1)]$ when it is used for the computation of DCT of $N/2$ point or a lower size, otherwise, the input is reset by an array of $(N/2)$ AND gates to disable this $(N/2)$-point DCT unit. The output of this $(N/2)$-point DCT unit is multiplexed with that of the OAU, which is preceded by the SAUs and IAU of the structure. The $N$ AND gates before IAU are used to disable the IAU, SAU, and OAU when the architecture is used to compute $(N/2)$-point DCT computation or a lower size. The input of the control unit, $mN$ is used to decide the size of DCT computation. Specifically, for $N = 32$, $m32$ is a 2-bits signal that is set to {00}, {01}, {10}, and {11} to compute four-, eight-, 16-, and 32-point DCT, respectively. The control unit generates sel 1 and sel 2, where sel 1 is used as control signals of $N$ MUXes and input of $N$ AND gates before IAU. sel 2 is used as the input $m(N/2)$ to two lower size reusable integer DCT units in a recursive manner.

The combinational logics for control units are shown in Fig. 4(b) and (c) for $N = 16$ and 32, respectively. For $N = 8$, $m8$ is a 1-bit signal that is used as sel 1 while sel 2 is not required since four point DCT is the smallest DCT. The proposed structure can compute one 32-point DCT, two 16-point DCTs, four eight point DCTs, and eight four-point DCTs, while the throughput remains the same as 32 DCT coefficients per cycle irrespective of the desired transform size.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

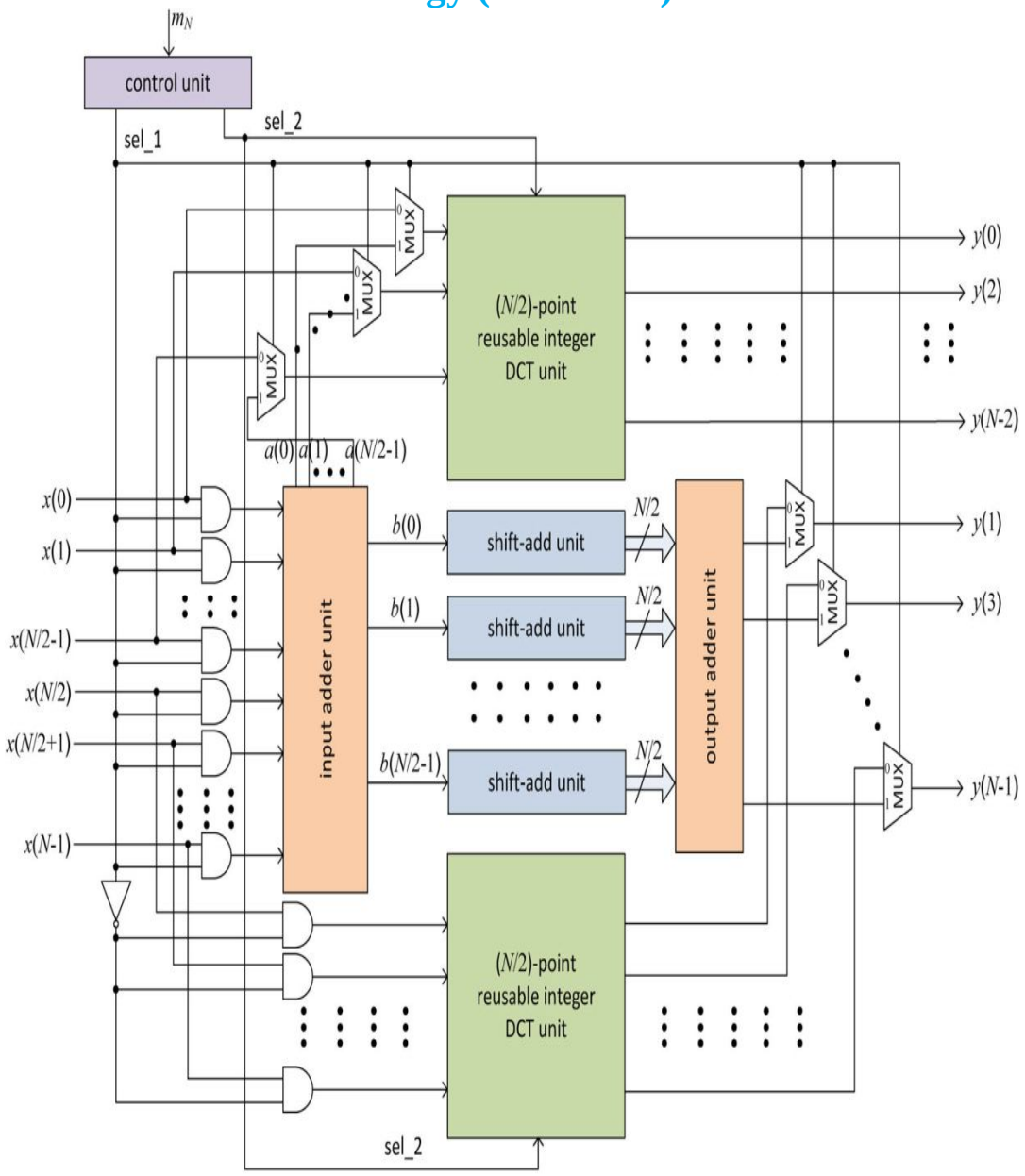



Fig. 6 Proposed reusable architecture for N = 8.

*E. Folded Structure for 2-D Integer DCT*

The folded structure for the computation of ($N \times N$)-point 2-D integer DCT is shown in Fig. 5(a). It consists of one $N$-point 1-D DCT module and a transposition buffer. The structure of the proposed 4 × 4 transposition buffer is shown in Fig.. It consists of 16 registers arranged in four rows and four columns. ($N \times N$) transposition buffer can store $N$ values in any one column of registers by enabling them by one of the enable signals EN$i$ for $i = 0, 1, \cdot\ \cdot\ \cdot, N-1$. One can select the content of one of the rows of registers through the MUXes. During the first $N$ successive cycles, the DCT module receives the successive columns of ($N \times N$) block of input for the computation of STAGE-1, and stores the intermediate results in the registers of successive columns in the transposition buffer. In the next $N$ cycles, contents of successive rows of the transposition buffer are selected by the MUXes and fed as input to the 1-D DCT module. $N$ MUXes are used at the input of the 1-D DCT module to select either the columns from the input buffer (during the first $N$ cycles) or the rows from the transposition buffer (during the next $N$ cycles).

Volume 5 Issue VI, June 2017
ISSN: 2321-9653

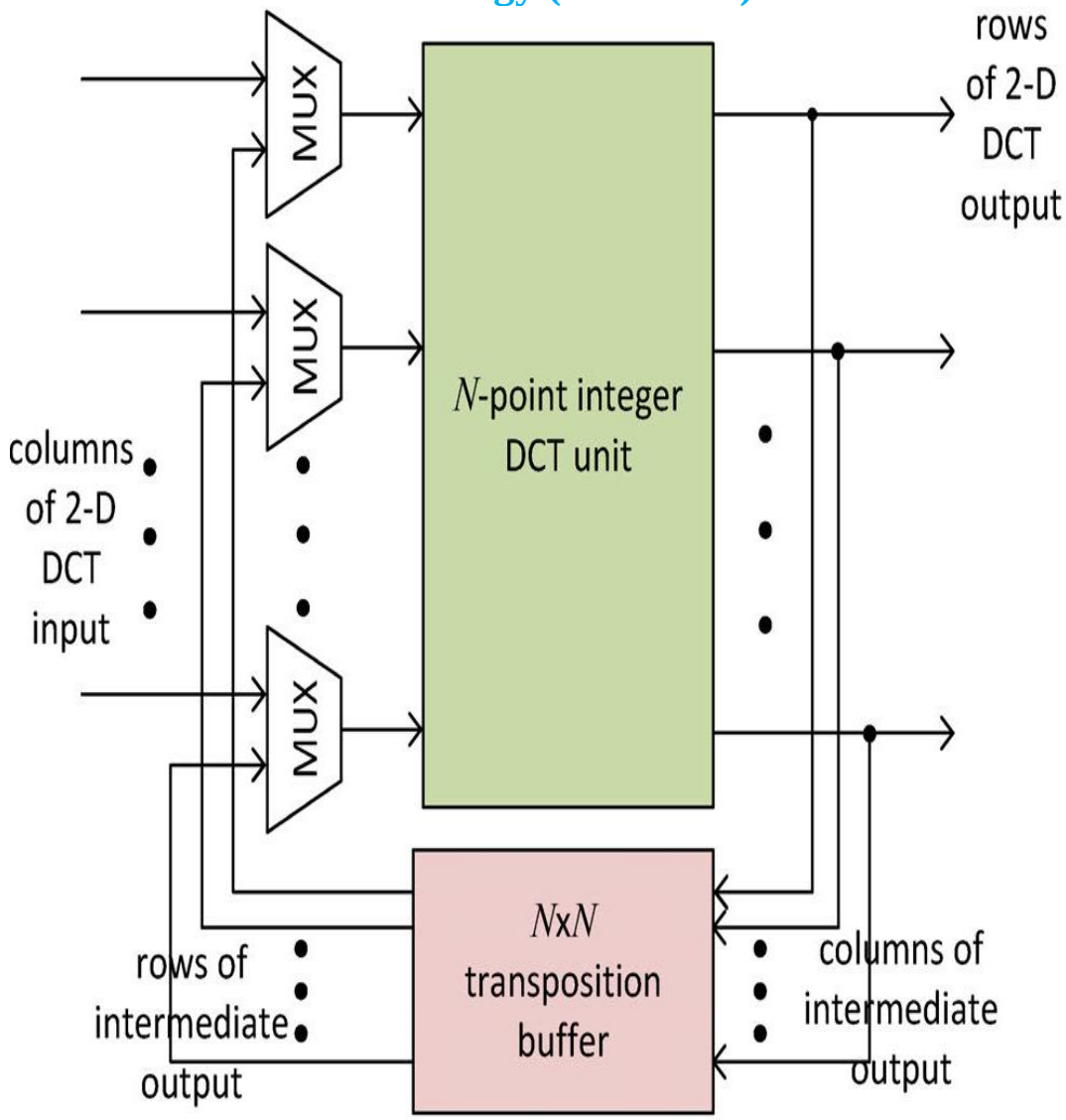




Fig.7 Folded Structure for 2-D Integer DCT

*F. Full-parallel structure for 2-d integer dct*

The full-parallel structure for ($N \times N$)-point 2-D integer DCT is shown in Fig.. It consists of two $N$-point 1-D DCT modules and a transposition buffer. The structure of the 4 × 4 transposition buffer for full-parallel structure is shown in Fig. 6(b). It consists of 16 register cells (RC) [shown in Fig. 6(c)] arranged in four rows and four columns. $N \times N$ transposition buffer can store $N$ values in a cycle either row wise or column-wise by selecting the inputs by the MUXes at the input of RCs. The output from RCs can also be collected either row-wise or column-wise. To read the output from the buffer, $N$ number of $(2N-1)$:1 MUXes [shown in Fig. 6(d)] are used, where outputs of the $i$th row and the $i$th column of RCs are fed as input to the $i$th MUX. For the first $N$ successive cycles, the $i$th MUX provides output of $N$ successive RCs on the $i$th row. In the next $N$ successive cycles, the $i$th MUX provides output of $N$ successive RCs on the $i$th column. By this arrangement, in the first $N$ cycles, we can read the output of $N$ successive columns of RCs and in the next $N$ cycles, we can read the output of $N$ successive rows of RCs. The transposition buffer in this case allows both read and write operations concurrently. If for the $N$ cycles, results are read and stored column-wise now, then in the next $N$ successive cycles, results are read and stored in the transposition buffer row-wise. The first 1-D DCT module receives the inputs column-wise from the input buffer.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653

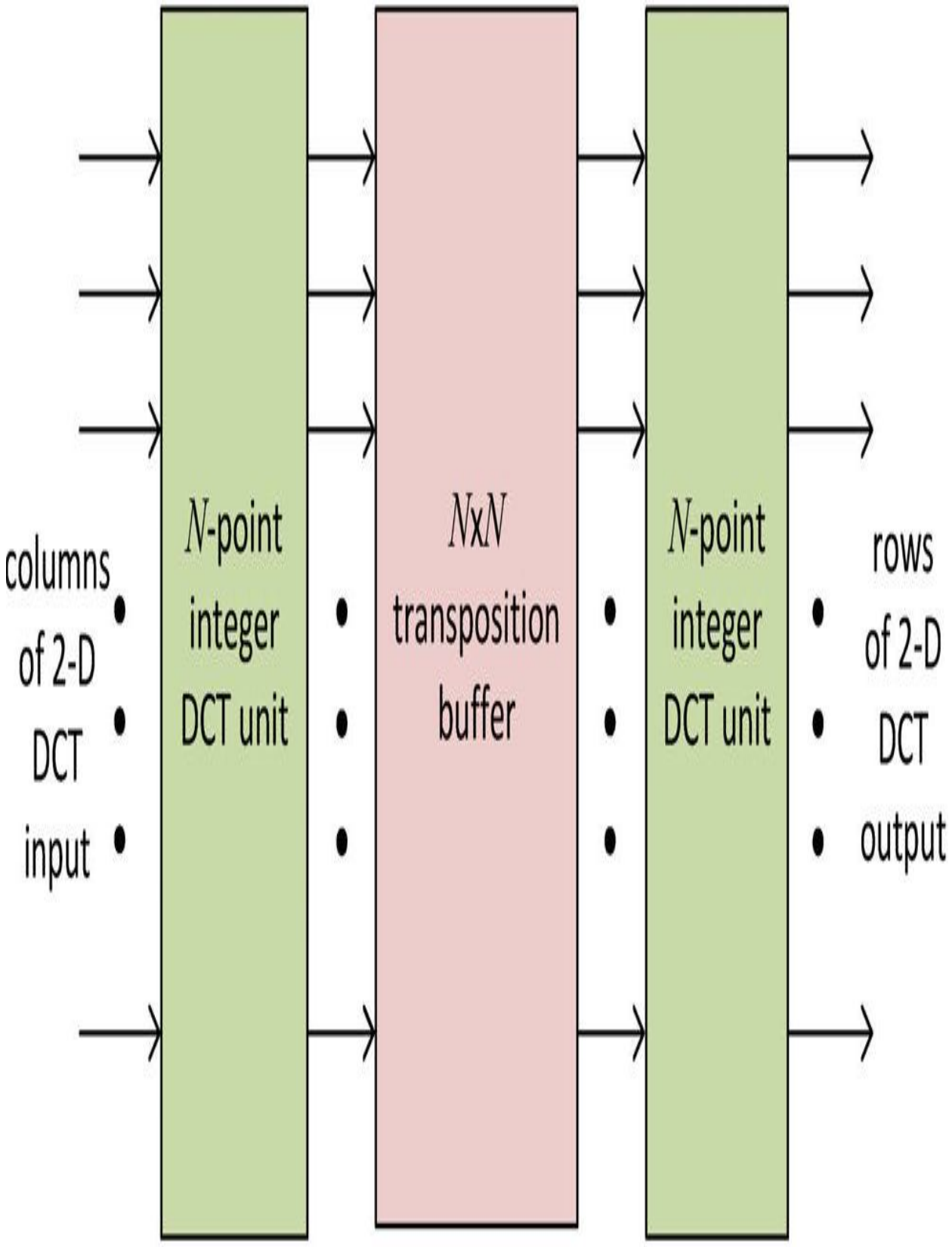
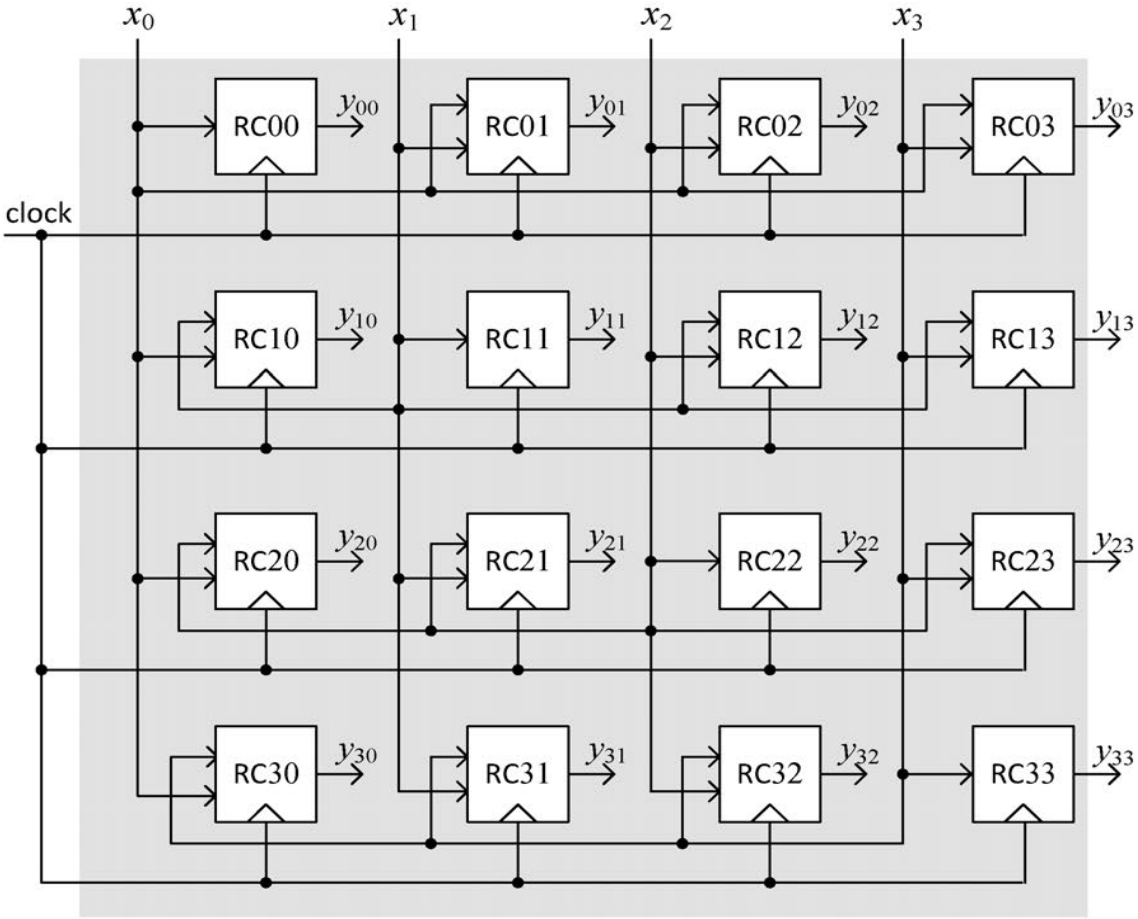




Fig. 8 Full-Parallel Structure for 2-D Integer DCT

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)




## V. EXPERIMENTAL SETUP AND RESULTS

*A. Verification Tool*

*1)* ModelSim 6.4c

*B. Synthesis Tool*

*1)* Xilinx ISE 13.2

*C. Modelsim*

*1)* *ModelSim SE - High Performance Simulation and Debug::* ModelSim SE is our UNIX, Linux, and Windows-based simulation and debug environment, combining high performance with the most powerful and intuitive GUI in the industry.

*2)* *High-Performance, Scalable Simulation Environment:* ModelSim provides seamless, scalable performance and capabilities. Through the use of a single compiler and library system for all ModelSim configurations, employing the right ModelSim configuration for project needs is as simple as pointing your environment to the appropriate installation directory.
ModelSim also supports very fast time-tenet-simulation turnarounds while maintaining high performance with its new black box use model, known as bbox. With bbox, non-changing elements can be compiled and optimized once and reused when running a modified version of the testbench. bbox delivers dramatic throughput improvements of up to 3X when running a large suite of testcases

*3)* *Easy-to-Use Simulation Environment:* An intelligently engineered graphical user interface (GUI) efficiently displays design data for analysis and debug. The default configuration of windows and information is designed to meet the needs of most users. However, the flexibility of the ModelSim SE GUI allows users to easily customize it to their preferences. The result is a feature-rich GU that is easy to use and quickly mastered. A message viewer enables simulation messages to be logged to the

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

ModelSim results file in addition to the standard transcript file. The GUI's organizational and filtering capabilities allow design and simulation information to be quickly reduced to focus on areas of interest, such as possible causes of design bugs.

ModelSim SE allows many debug and analysis capabilities to be employed post-simulation on saved results, as well as during live simulation runs. For example, the coverage viewer analyzes and annotates source code with code coverage results, including FSM state and transition, statement, expression, branch, and toggle coverage. Signal values can be annotated in the source window and viewed in the waveform viewer. Race conditions, delta, and event activity can be analyzed in the list and wave windows. User-defined enumeration values can be easily defined for quicker understanding of simulation results. For improved debug productivity, ModelSim also has graphical and textual dataflow capabilities. The memory window identifies memories in the design and accommodates flexible viewing and modification of the memory contents. Powerful search, fill, load, and save functionalities are supported. The memory window allows memories to be pre-loaded with specific or randomly generated values, saving the time-consuming step of initializing sections of the simulation merely to load memories. All functions are available via the command line, so they can be used in scripting.

*D. Xilinx Ise*

1) *Introduction:* For two-and-a-half decades, Xilinx has been at the forefront of the programmable logic revolution, with the invention and continued migration of FPGA platform technology. During that time, the role of the FPGA has evolved from a vehicle for prototyping and glue-logic to a highly flexible alternative to ASICs and ASSPs for a host of applications and markets. Today, Xilinx® FPGAs have become strategically essential to world-class system companies that are hoping to survive and compete in these times of extreme global economic instability, turning what was once the programmable revolution into the "programmable imperative" for both Xilinx and our customers.

2) *XILINX ISE Design Tools:* Xilinx ISE is the design tool provided by Xilinx. Xilinx would be virtually identical for our purposes.

There are four fundamental steps in all digital logic design. These consist of:

*a) Design:* The schematic or code that describes the circuit.

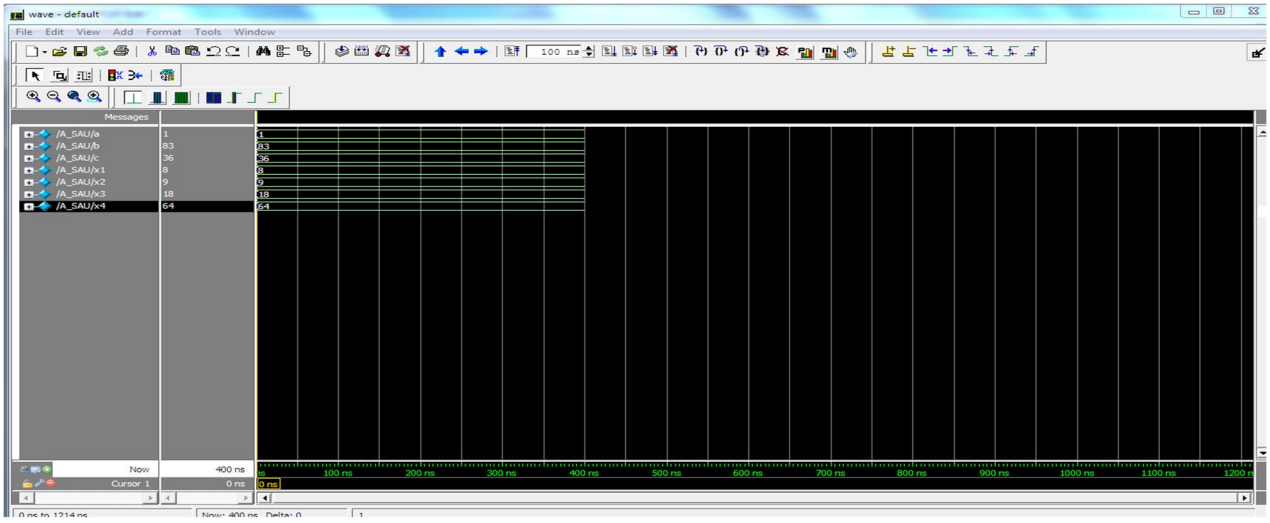
*b) Synthesis:* The intermediate conversion of human readable circuit description to FPGA code (EDIF) format. It involves syntax checking and combining of all the separate design files into a single file.

*c) Place & Route:* Where the layout of the circuit is finalized. This is the translation of the EDIF into logic gates on the FPGA.

*d) Program:* The FPGA is updated to reflect the design through the use of programming (.bit) files.Test bench simulation is in the second step. As its name implies, it is used for testing the design by simulating the result of driving the inputs and observing the outputs to verify your design.

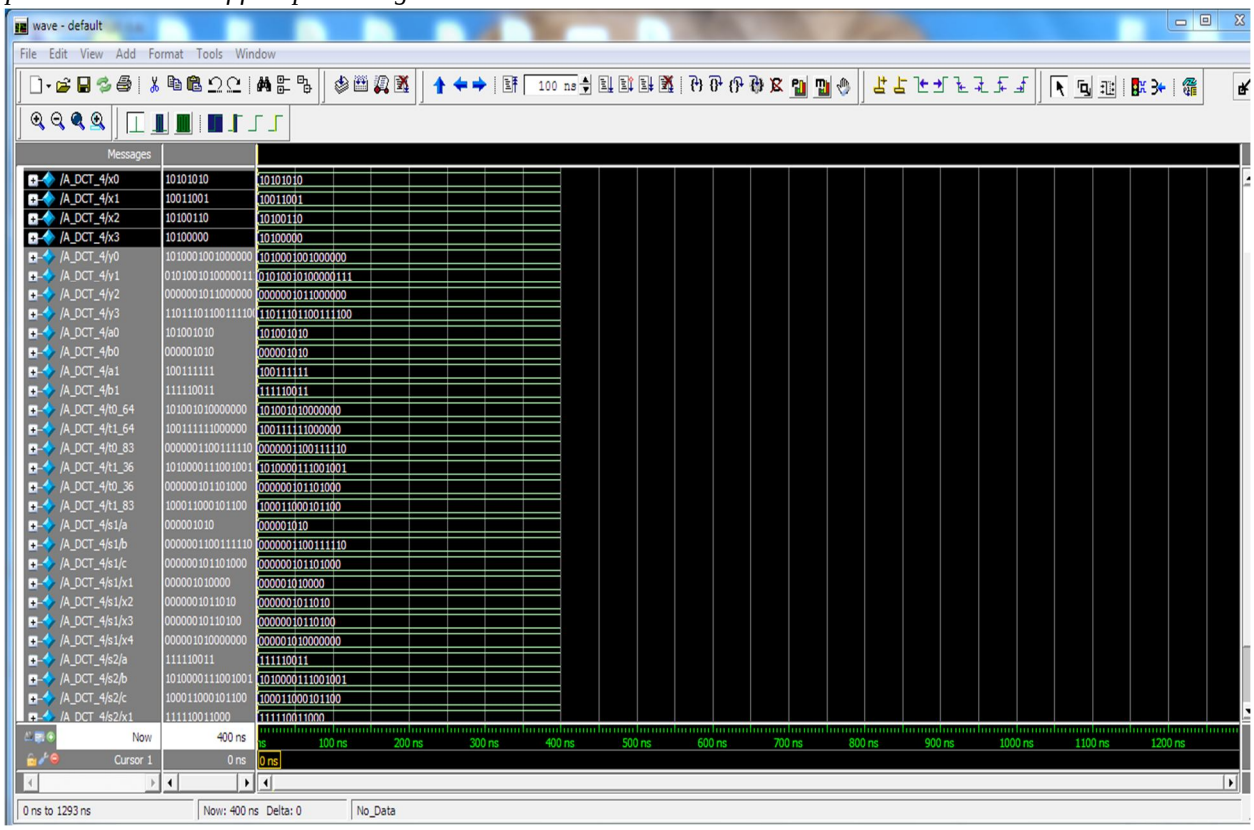
ISE has the capability to do a variety of different design methodologies including: Schematic Capture, Finite State Machine and Hardware Descriptive Language (VHDL or Verilog).

3) *Structure of SAU:*

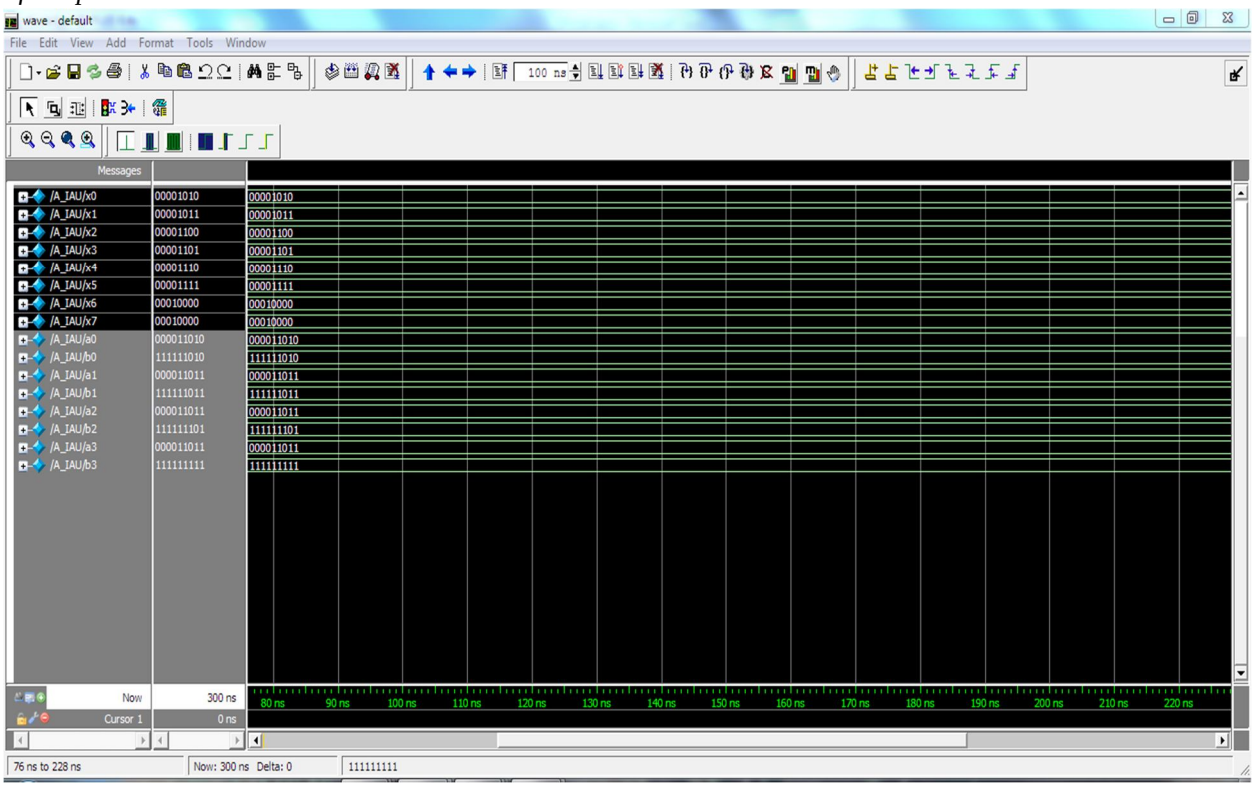


Volume 5 Issue VI, June 2017
ISSN: 2321-9653


*4) Proposed architecture of four-point integer DCT*

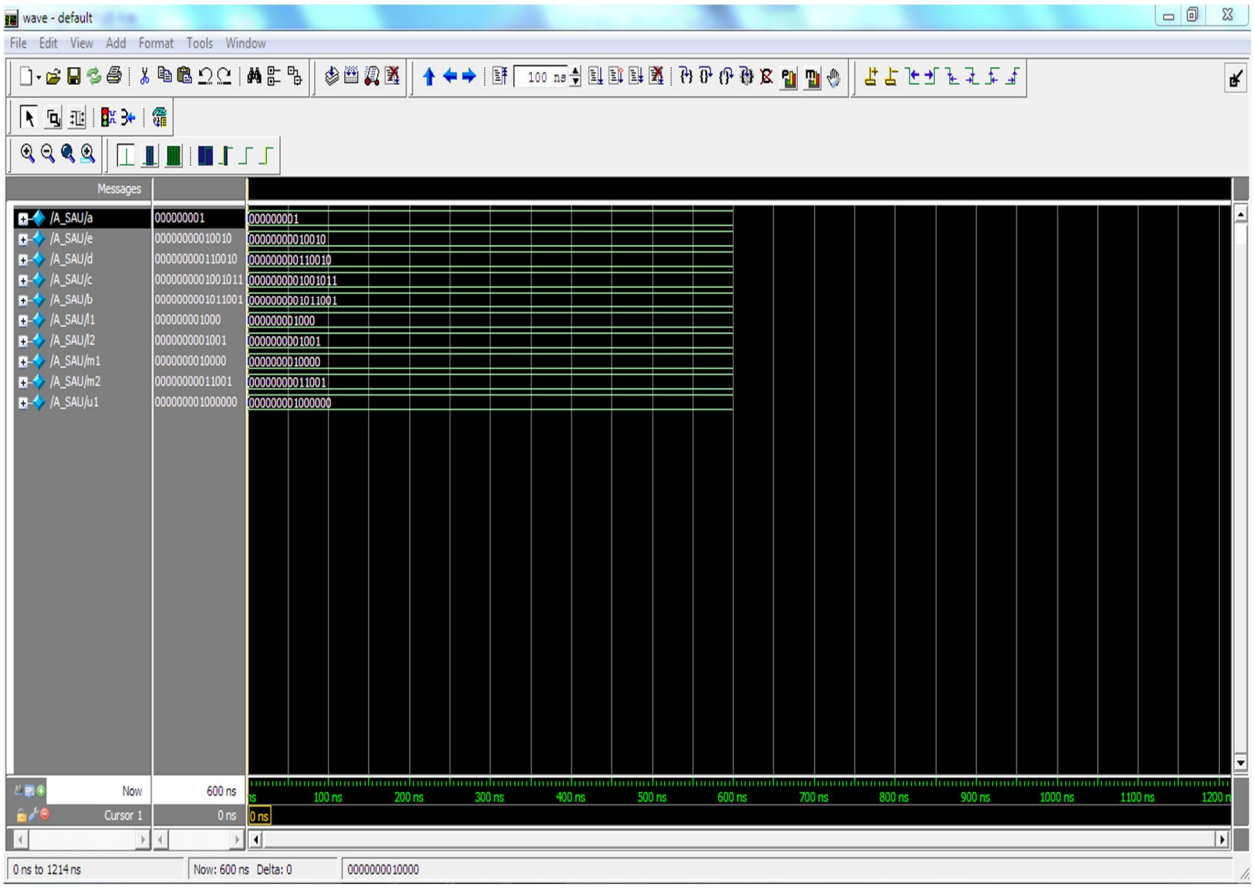



*5) IAU for 8 points*

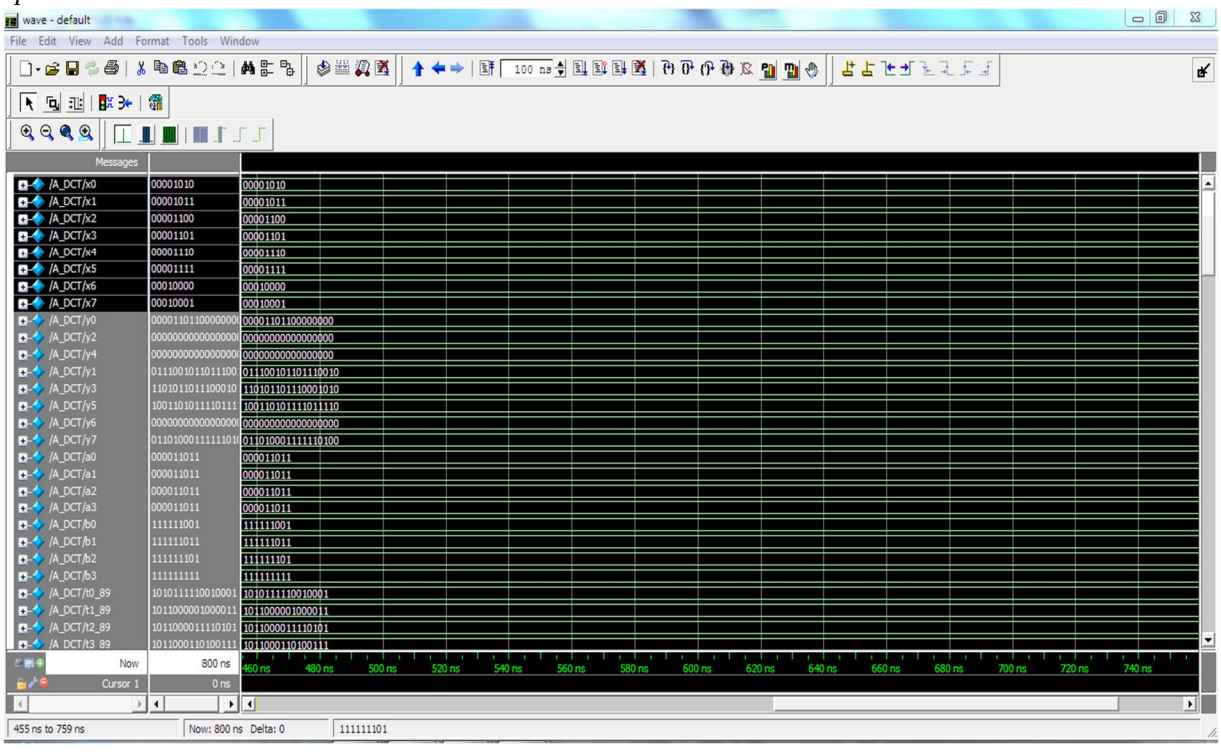


Volume 5 Issue VI, June 2017
ISSN: 2321-9653


*6) SAU for 8 points*

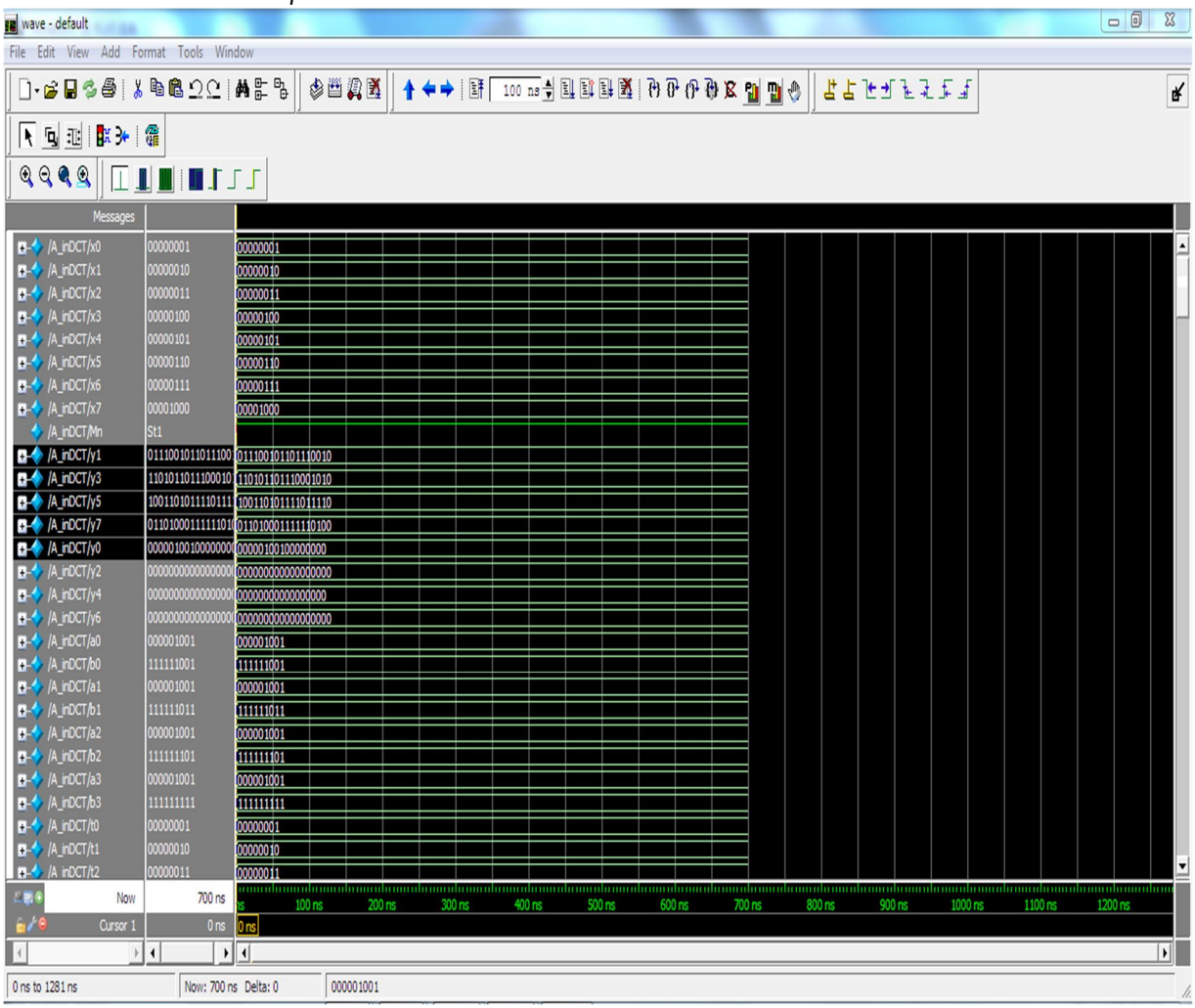



*7) Eight point DCT*

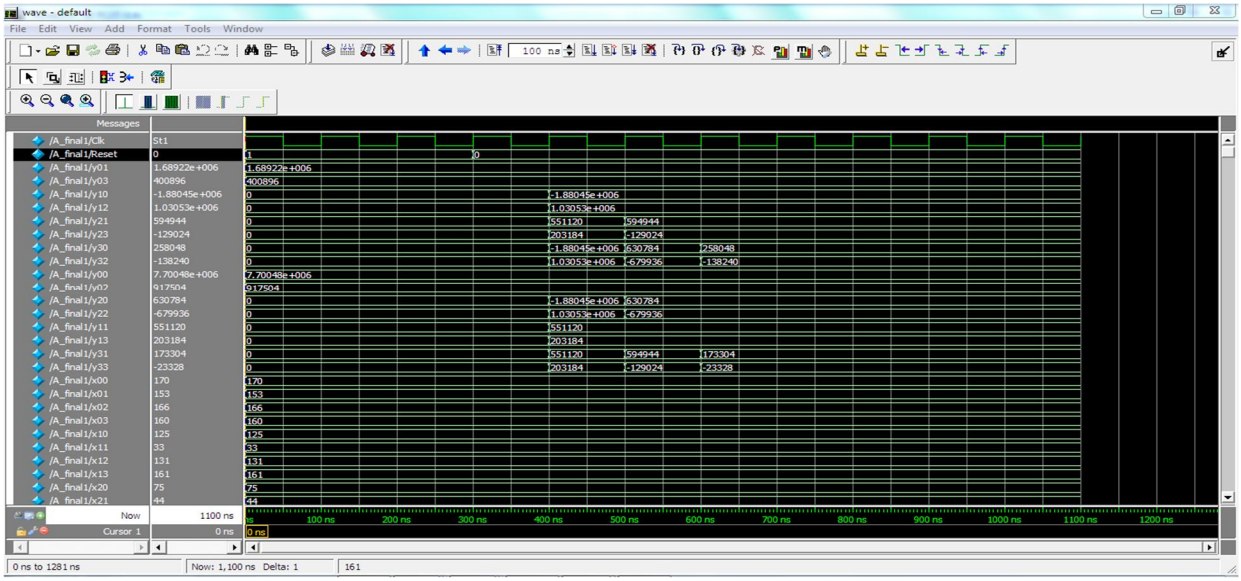


Volume 5 Issue VI, June 2017
ISSN: 2321-9653


*8) Proposed reusable architecture for N = 8*

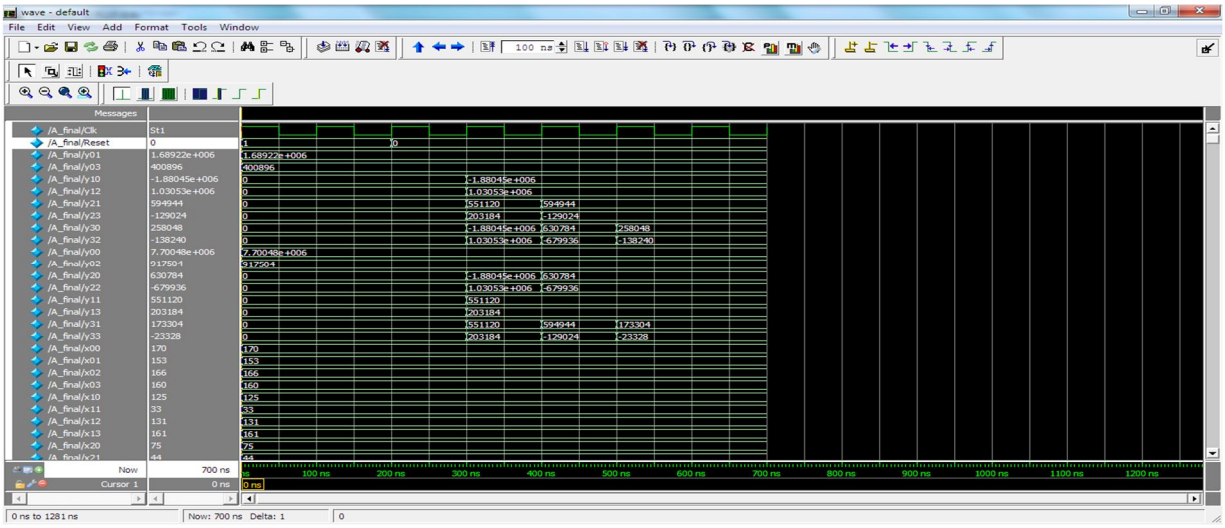



*9) Folded Structure*



Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

*10) Full Parallel*




*E. Synthesis Report*

*1) Device Utilization Summary*

*a) Folded Structure*

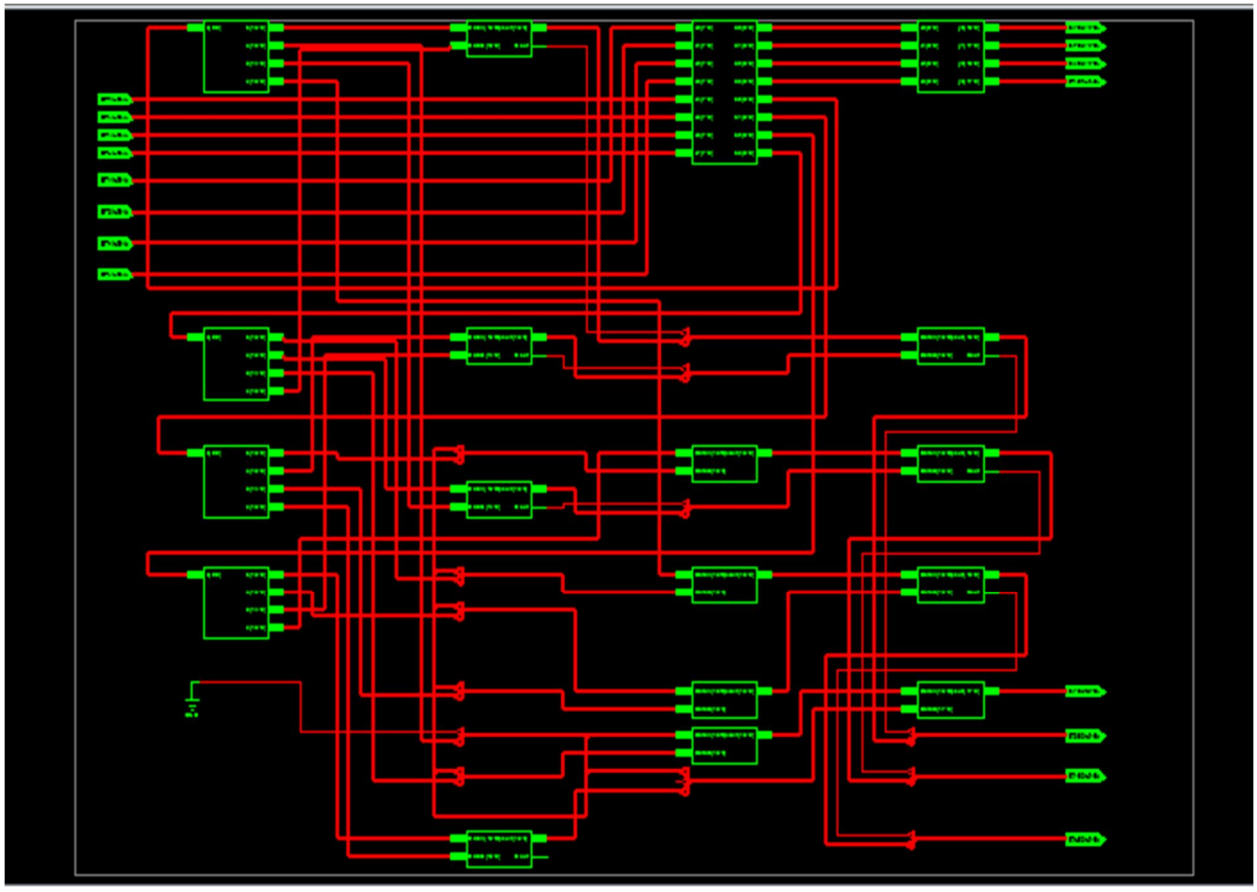
| Device Utilization Summary | | | | |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 9 | 7,168 | 1% | |
| Number of 4 input LUTs | 562 | 7,168 | 7% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 380 | 3,584 | 10% | |
| Number of Slices containing only related logic | 380 | 380 | 100% | |
| Number of Slices containing unrelated logic | 0 | 380 | 0% | |
| **Total Number of 4 input LUTs** | 609 | 7,168 | 8% | |
| Number used as logic | 562 | | | |
| Number used as a route-thru | 47 | | | |
| Number of bonded IOBs | 370 | 141 | 262% | OVERMAPPED |
| Number of GCLKs | 1 | 8 | 12% | |
| **Total equivalent gate count for design** | 7,464 | | | |
| Additional JTAG gate count for IOBs | 17,760 | | | |

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

*b) Full Parallel*

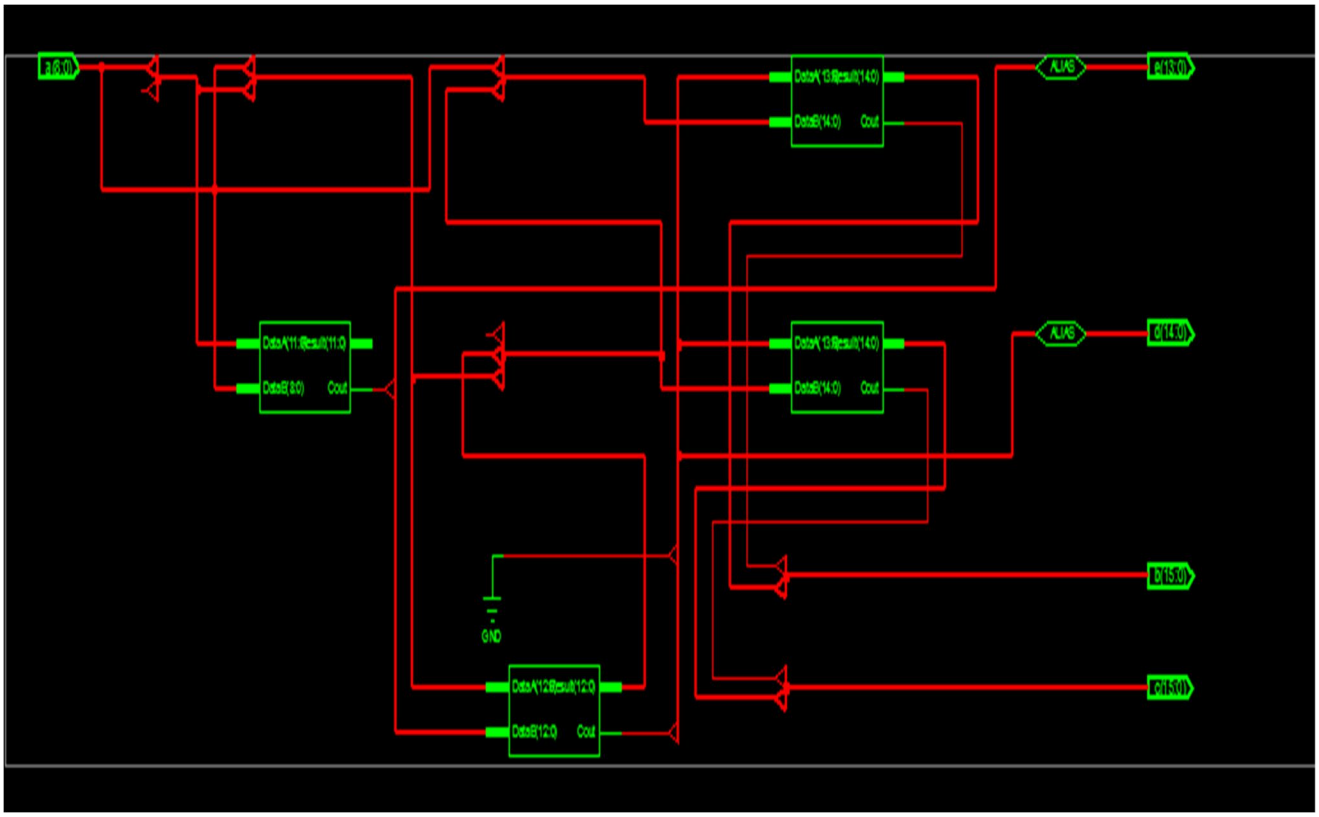
| Device Utilization Summary | | | | |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 9 | 7,168 | 1% | |
| Number of 4 input LUTs | 510 | 7,168 | 7% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 364 | 3,584 | 10% | |
| Number of Slices containing only related logic | 364 | 364 | 100% | |
| Number of Slices containing unrelated logic | 0 | 364 | 0% | |
| **Total Number of 4 input LUTs** | 556 | 7,168 | 7% | |
| Number used as logic | 510 | | | |
| Number used as a route-thru | 46 | | | |
| Number of bonded IOBs | 370 | 141 | 262% | OVERMAPPED |
| Number of GCLKs | 1 | 8 | 12% | |
| **Total equivalent gate count for design** | 6,999 | | | |
| Additional JTAG gate count for IOBs | 17,760 | | | |

*c) Rtl Schematic*

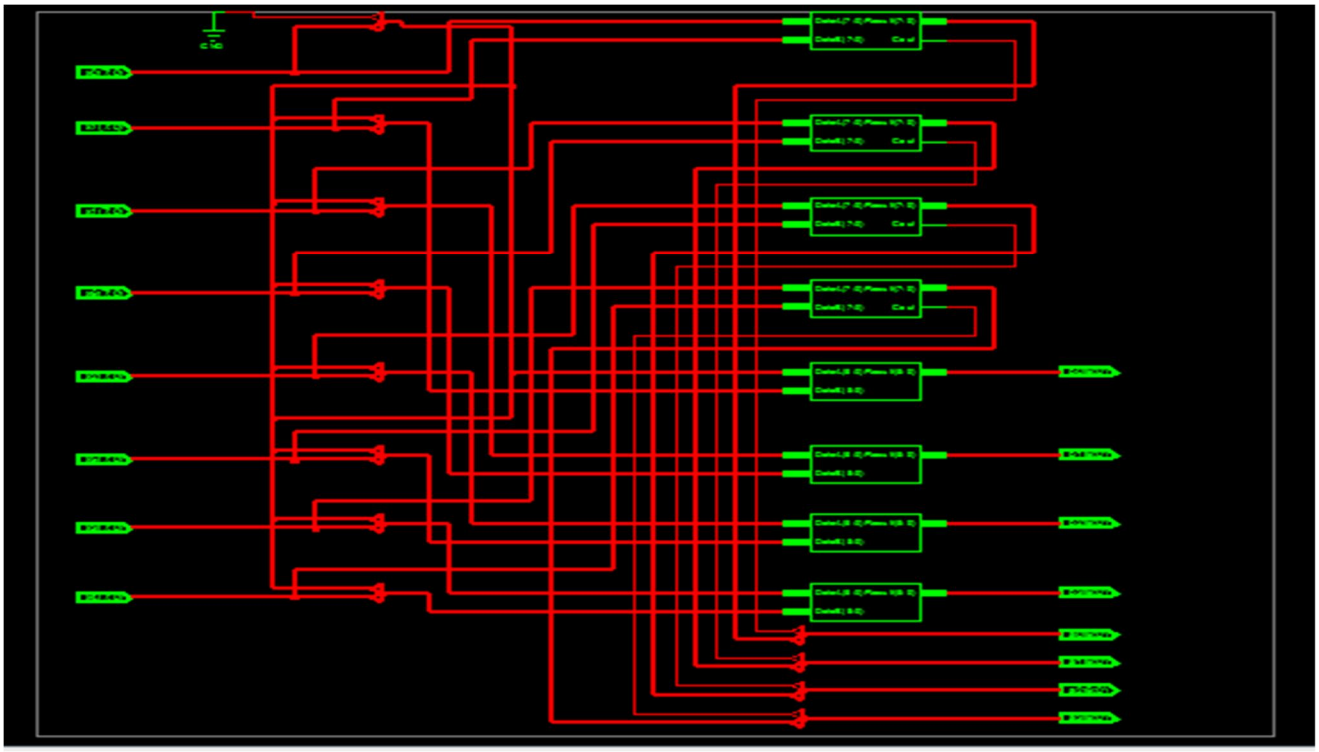


Volume 5 Issue VI, June 2017
ISSN: 2321-9653


*d) Sau*



*e) Iau*



Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

## VI. CONCLUSION

we have proposed area- and power-efficient architectures for the implementation of integer DCT of different lengths to be used in HEVC. The computation of *N*-point 1-D DCT involves an (*N*/2)-point 1-D DCT and a vector-matrix multiplication with a constant matrix of size (*N*/2) × (*N*/2). We have shown that the MCM-based architecture is highly regular and involves significantly less area-delay complexity and less energy consumption than the direct implementation of matrix multiplication for odd DCT coefficients. We have used the proposed architecture to derive a reusable architecture for DCT that can compute the DCT of lengths 4, 8, 16, and 32 with throughput of 32 output coefficients per cycle. We have also shown that proposed design could be pruned to reduce the area and power complexity of forward DCT without significant loss of quality. We have proposed power-efficient architectures for folded and full-parallel implementations of 2-D DCT, where no data movement takes place within the transposition buffer. From the synthesis result, it is found that the proposed algorithm with pruning involves nearly 30% less ADP and 35% less EPS compared to the reference algorithm in average for integer DCT of lengths 4, 8, 16, and 32 with nearly the same throughput rate.

## REFERENCES

[1] N. Ahmed, T. Natarajan, and K. Rao, “Discrete cosine transform,” IEEE Trans. Comput., vol. 100, no. 1, pp. 90–93, Jan. 1974.

[2] W. Cham and Y. Chan, “An order-16 integer cosine transform,” IEEE Trans. Signal Process., vol. 39, no. 5, pp. 1205–1208, May 1991.

[3] Y. Chen, S. Oraintara, and T. Nguyen, “Video compression using integer DCT,” in Proc. IEEE Int. Conf. Image Process., Sep. 2000, pp. 844–845.

[4] J. Wu and Y. Li, “A new type of integer DCT transform radix and its rapid algorithm,” in Proc. Int. Conf. Electric Inform. Control Eng., Apr. 2011, pp. 1063–1066.

[5] A. M. Ahmed and D. D. Day, “Comparison between the cosine and Hartley based naturalness preserving transforms for image watermarking and data hiding,” in Proc. First Canad. Conf. Comput. Robot Vision, May 2004, pp. 247–251.

[6] M. N. Haggag, M. El-Sharkawy, and G. Fahmy, “Efficient fast multiplication-free integer transformation for the 2-D DCT H.265 standard,” in Proc. Int. Conf. Image Process., Sep. 2010, pp. 3769–3772.

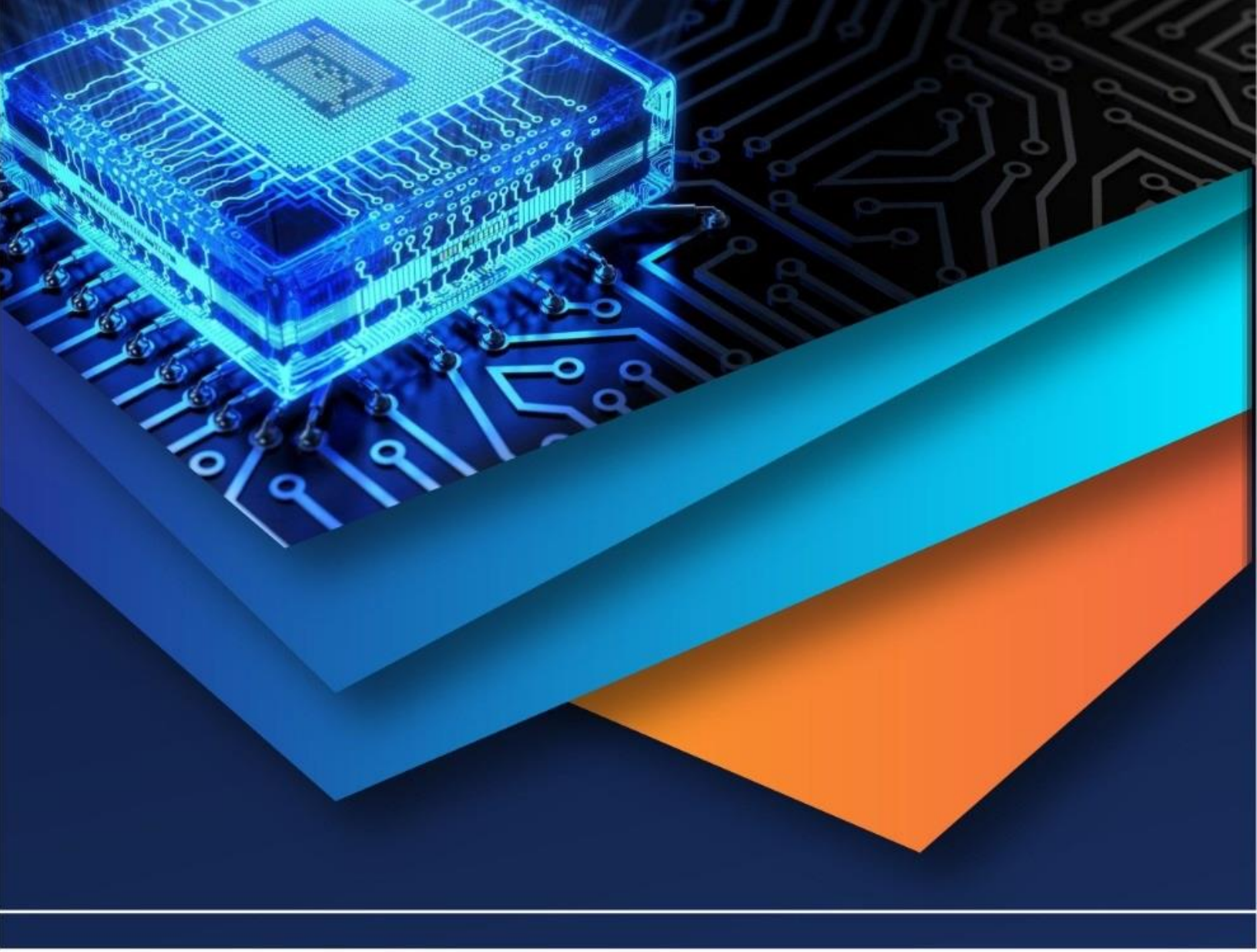
[7] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS and Consent), JCT-VC L1003, Geneva, Switzerland, Jan. 2013.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.

[9] .A. Ahmed, M. U. Shahid, and A. Rehman, “N Point DCT VLSI Architecture for Emerging HEVC Standard,” in Proc. VLSI Design, vol. 2012, Article 752024, pp. 1–13, 2012.

[10] S. Shen, W. Shen, Y. Fan, and X. Zeng, “A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards,” in Proc. IEEE Int. Conf. Multimedia Expo, Jul. 2012, pp. 788–793.

[11] J.-S. Park, W.-J. Nam, S.-M. Han, and S. Lee, “2-D large inverse transform (16x16, 32x32) for HEVC (high efficiency video coding),” J. Semicond. Technol. Sci., vol. 12, no. 2, pp. 203–211, Jun. 2012.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY