Novel Low Power, High Speed Hardware Implementation of 1D DCT/IDCT Using Xilinx FPGA

Rajesh Kannan Megalingam, Vineeth Sarma V., Venkat Krishnan B., Mithun M., Rahul Srikumar
Department of Electronics and Communication
Amrita School of Engineering, Amrita Vishwa Vidyapeetham
Kollam, Kerala, India.
rajeshm@amritapuri.amrita.edu, vineethisalways4u@gmail.com, venkatkrishnan.b@gmail.com, mithun34@ieee.org, rahul44@ieee.org

Abstract— **DCT/IDCT finds potent application in the field of image and signal processing. In this paper we concentrate on a novel five stage pipelined implementation, which consumes less power. The design uses Verilog HDL and is simulated in Modelsim 6.3b. Matlab is used to generate the data in binary format which serves as the input data and cosine values for computing 1D DCT/IDCT in HDL. There are other low power implementations as in [4], but in this novel implementation we prove that a lower power implementation can be done which also increases speed (of what?) by approximately five times over that of conventional implementations. The implementation of both non-pipelined (conventional) and pipelined method uses Xilinx XC3S4000 FPGA. The DCT/IDCT is found using the most common and optimum method of taking inputs as set of eight data elements [1], [2]. Finally, a comparison of the speeds of both implementations is made, and the speed up achieved by the low power pipelined implementation of 1D-DCT/IDCT is presented.**

Keywords- DCT/IDCT , pipeline, FPGA

I. INTRODUCTION

DCT is commonly used in data compression applications internationally due to its good de-correlation properties [2]. DCT/IDCT finds an application in JPEG, MPEG, H.261 and H.263 image and video compression standards. Its widespread use can be attributed to the energy compaction quality of the cosine transform. There are a number of implementations of 1D DCT/IDCT, but in this paper we are focusing on a low power - high speed implementation of 1D-DCT/IDCT. 1D IDCT is also widely used in image processing. Image processing employs transform coding where the transform can be cosine (transform) or Wavelet based, with the cosine transform being the more popular in recent times. Transform coding is based on the premise that the pixels in an image exhibit a certain level of correlation with their neighboring pixels. Consequently, this correlation can be exploited to predict the value of a pixel from its respective neighbour. A transformation is, therefore, defined that maps this spatial (correlated) data into transformed (uncorrelated) coefficients. Clearly, the transformation should utilize the fact that the information content of an individual pixel is relatively small, i.e., the fact that, to a large extent, the visual contribution of a pixel can be predicted using its neighbours [2].

II. ONE DIMENSIONAL DCT

A discrete cosine transform helps in expressing a finite number of data points as the sum of cosine functions that oscillate at different frequencies. 1D-DCT is given by the equation:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x)\cos[\pi(2x+1)u/2N] \quad (1)$$

For u=0, 1, 2, 3….. (N-1). And in a similar way the 1-D IDCT is defined as:

$$f(x) = \alpha(u) \sum_{u=0}^{N-1} \alpha(u)\, C(u) \cos[\pi(2x+1)u/2N] \quad (2)$$

For x = 0, 1, 2, 3…. (N-1) and for both the equations

$$\alpha(u) = (1/N)^{0.5} \quad \text{for } u=0 \quad (3)$$

$$(2/N)^{0.5} \quad \text{for } u \neq 0$$

The DCT of data can be found directly given the data values, but if the data is large say about 100 or 500, it is not possible to find the DCT/IDCT of the entire data. Hence, we generally divide the given data elements into sets of eight [1][6]. The normal implementations of 1D-DCT uses multipliers and adder-subtractors in the design as given in [1]. The 1D-DCT of a data set of eight elements can be found using the matrix shown below.

$$\begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} c4 & c4 & c4 & c4 & c4 & c4 & c4 & c4 \\ c1 & c3 & c5 & c7 & -c7 & -c5 & -c3 & -c1 \\ c2 & c4 & c6 & -c2 & -c2 & -c6 & c6 & c2 \\ c3 & -c7 & -c1 & -c5 & -c5 & c1 & c7 & -c3 \\ c5 & -c1 & c7 & c3 & -c3 & -c7 & c1 & -c5 \\ c6 & -c2 & c2 & -c6 & -c6 & c2 & -c2 & c6 \\ c7 & -c5 & c3 & -c1 & c1 & -c3 & c5 & -c7 \end{bmatrix} \begin{bmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{bmatrix}$$

Figure 1. Matrix method of implementing 1D-DCT/IDCT

978-0-7695-3892-1/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCTD.2009.194

Where x(0) to x(7) represent**S** the eight data elements and Cx= cos(πx/16). This implementation requires the use of 64 multipliers for multiplying the data values with the cosine values, 56 adders for adding the result after multiplication, and shifters that perform arithmetic right shifts to divide by two. The major concern is the number of adders and multipliers that are used and the power consumed by the design. This matrix can be further simplified as in [1], (which focuses more on) reducing the number of adders and multipliers to decrease the net power consumed by the design and to decrease the number of gates in the design. The simplified implementation is also suggested in [1] and the matrix is shown in "Fig. 2".

This implementation reduces the number of multipliers to 32 and the number of adders to 24, since 32 of the coefficients of the reduced matrix are zeros. Comparing this matrix implementation with previous implementations we see that the number of multipliers is reduced by half (from 64 to 32), and the number of adders is reduced by more than half (from 56 to 24). Consequently, this design consumes less power compared to implementation of the matrix in "Fig.1". Our implementation of 1D DCT/IDCT is similar to the matrix implementation. The matrix implementation has been proven to consume less power, but when we added a five stage pipeline its speed was increased. In fact, we have compared the speed of both pipelined and non-pipelined implementations of DCT/IDCT to estimate the speed-up that can be achieved by the pipelined implementations of DCT/IDCT. The speed up thereby attained also depends upon the number of data elements to be processed. As the number of data elements increases the speed up attained approaches five, which means the pipelined implementation can perform the DCT/IDCT operation five times faster than the conventional implementation as suggested by [1].

"Fig.3" shows a butterfly representation of the 1D DCT/IDCT. Here x0 to x7 represents the input data. The input data is first added or subtracted and then multiplied with the cosine values. The result after multiplying is then added between them and then divided by two which gives the final output y0 to y7 of 1D-DCT/IDCT.

$$\begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} c4 & c4 & c4 & c4 & 0 & 0 & 0 & 0 \\ c2 & c6 & -c6 & -c2 & 0 & 0 & 0 & 0 \\ c4 & -c4 & -c4 & c4 & 0 & 0 & 0 & 0 \\ c6 & -c2 & c2 & -c6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c1 & c3 & c5 & c7 \\ 0 & 0 & 0 & 0 & c3 & -c7 & -c1 & -c5 \\ 0 & 0 & 0 & 0 & c5 & -c1 & c7 & c3 \\ 0 & 0 & 0 & 0 & c7 & -c5 & c3 & -c1 \end{bmatrix} \begin{bmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \\ x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{bmatrix}$$

Figure 2. Matrix method which requires a reduced number of multipliers and adders

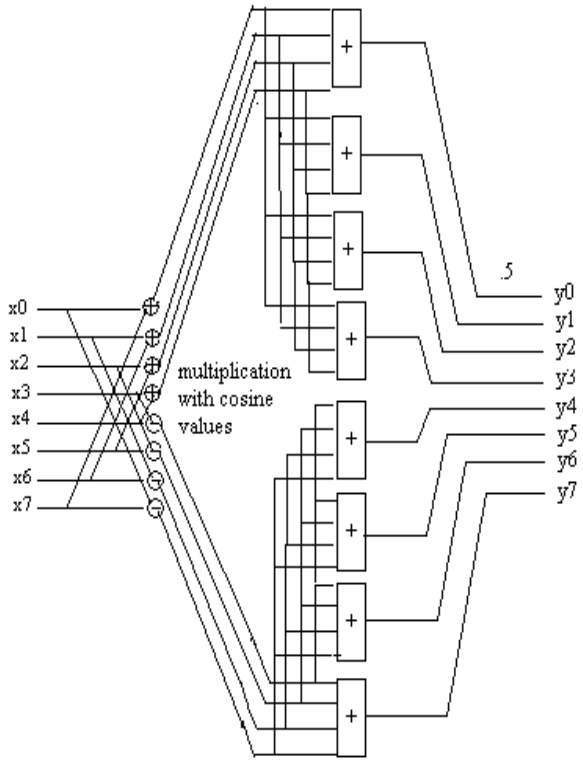

Figure 3. Butterfly representation of the matrix given in Figure 1.

III. FIVE STAGE PIPELINED 1D DCT IMPLEMENTATION

When we implemented the pipelined 1D DCT/IDCT, there was not much increase in hardware, but there was nearly a five times increase in speed over conventional implementation. From the matrix in "Fig.2", we identified five stages, which we have implemented as a five stage pipeline. The five stages that are taken for pipelining are,

- The FETCH stage represented by F involves the fetching of data from the memory to the local registers.
- The ADDSUB stage represented by AS involves adding or subtracting the data, e.g.: x(0) + x(7),.
- The MULTIPLY stage represented by M involves multiplication with the cosine values, e.g.: C_4 * [x(0) + x(7)],.
- The ADD stage represented by A involves addition to get the output of matrix multiplication.
- The DIVIDE stage represented by D involves division by 2 which is obtained by signed right shift or the arithmetic right shift of the data elements.

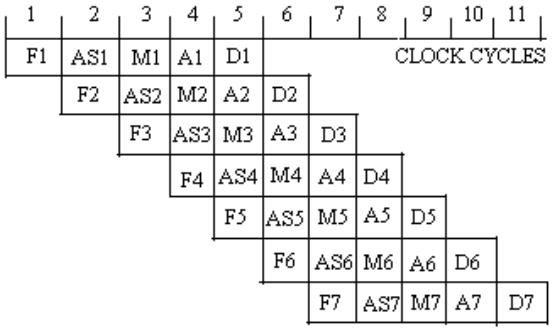

Figure 4. Represents the space-time diagram of pipelined Implementation (or A representation of the space-time diagram of pipelined Implementation)

Pipelining can be implemented by placing one register in between each stage, which is the only additional hardware required as shown in "Fig. 5". In this method, the first set of eight data elements are fetched in the first clock cycle. In the second clock cycle, when the data fetched in the first clock cycle are being added and subtracted, the second set of eight data elements are fetched into the registers. In the third clock cycle, the data set of eight elements, fetched in the first clock cycle, are multiplied with the cosine values. At the same time, the second set is added or subtracted, and the third set of data is fetched into the local registers, so on. In this implementation the 1D DCT output of the first set of eight data elements is obtained after the fifth clock cycle, and the second set of output after the sixth clock cycle. This procedure gives approximately one set of eight-element-output per clock cycle.

IV. FIVE STAGE PIPELINED 1D IDCT IMPLEMENTATION

The IDCT can also be implemented using the matrix method given in "Fig.2". This requires a modification **of** the matrix to be implemented which can be represented as in "Fig.6".The output when IDCT is implemented using the matrix in "Fig.6" is not x0, x1, etc. but their linear combinations such as x0+x7, x1-x6, etc. as shown in "Fig.6." x0 to x7 can be found out from the linear equations that are obtained after the inverse operation is performed, as shown in the matrix in "Fig. 6".

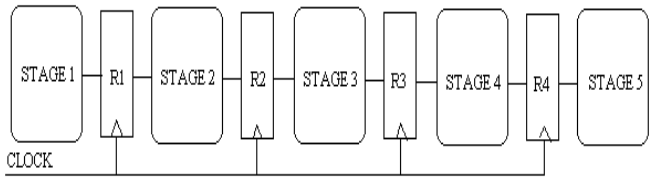

Figure 5. Representation of 5 stage pipeline

$$\begin{bmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \\ x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{bmatrix} = 2X \begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} \begin{bmatrix} c4 & c4 & c4 & c4 & 0 & 0 & 0 & 0 \\ c2 & c6 & -c6 & -c2 & 0 & 0 & 0 & 0 \\ c4 & -c4 & -c4 & c4 & 0 & 0 & 0 & 0 \\ c6 & -c2 & c2 & -c6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c1 & c3 & c5 & c7 \\ 0 & 0 & 0 & 0 & c3 & -c7 & -c1 & -c5 \\ 0 & 0 & 0 & 0 & c5 & -c1 & c7 & c3 \\ 0 & 0 & 0 & 0 & c7 & -c5 & c3 & -c1 \end{bmatrix}^{-1}$$

Figure 6. Matrix representing IDCT

For example if x0+x7= o1 and x0-x7=o2, then x0 is given by (o1+o2)/2 and x7 is given by (o1-o2)/2. In a similar manner all the x values could be found out. Here, in the case of IDCT we also implement a five stage pipeline. The five stages in the pipeline are as follows,

- The FETCH stage, represented by F involves fetching the data from the memory.
- The MULTIPLY stage represented by M involves multiplication with the cosine values. e.g., C_4' * [y0]. C_4' represents the inverse cosine value at x=4.
- The ADD stage represented by A, involves addition to get the output of matrix multiplication.
- The MULTIPLY by 2 stage represented by M' involves multiplication by 2 obtained by signed left shift or the arithmetic left shift of the data.
- The fifth stage represented by P involves finding x0, x1, x2, etc from the result obtained.

V. DESIGN AND IMPLEMENTATION

The hardware design of 1D DCT/IDCT uses Verilog HDL and the simulation uses Modelsim 6.3b. The cosine values represented as Cx in the section II is found using Matlab and it is made into a bit file. The input data is also given as bit file which is stored in the memory module and the result, after computation, is also stored in the memory module as shown in "Fig.7'.

The 1D-IDCT is found using a method similar to that used for finding the 1D-DCT. The only difference being that the inverse cosine values are given as inputs instead of cosine values, and that the operations performed on the data elements are different, as mentioned in the section IV. The flowchart of the implementations of DCT and IDCT are as shown in "Fig.7 and Fig. 8". The flowchart shows all phases of the implementations. The data input is made into a bit file using Matlab and the design uses Verilog HDL. The design is simulated using Modelsim 6.3b. If the output obtained is correct, it is implemented in hardware and if there is a mistake, then the code is rechecked. The correctness of the result can be verified by the results from the Matlab.

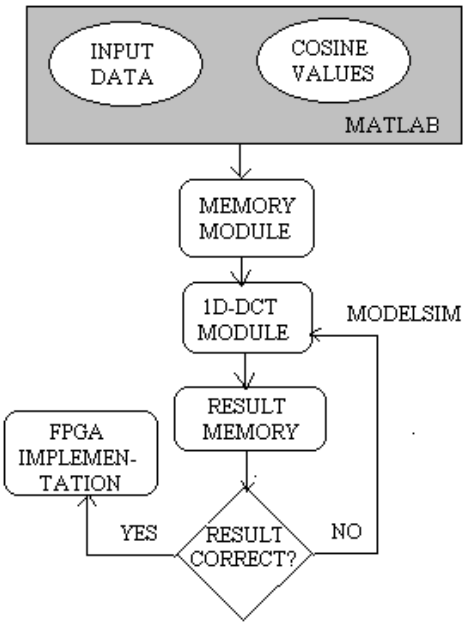

Figure 7. Implementation of 1D-DCT

VI. PERFORMANCE ANALYSIS

The two hardware implementations are for the pipelined and the non-pipelined. Higher performance was obtained in the case of pipelined implementation of DCT. We already saw that there are five operations that must be performed. Suppose the clock frequency is 10MHz

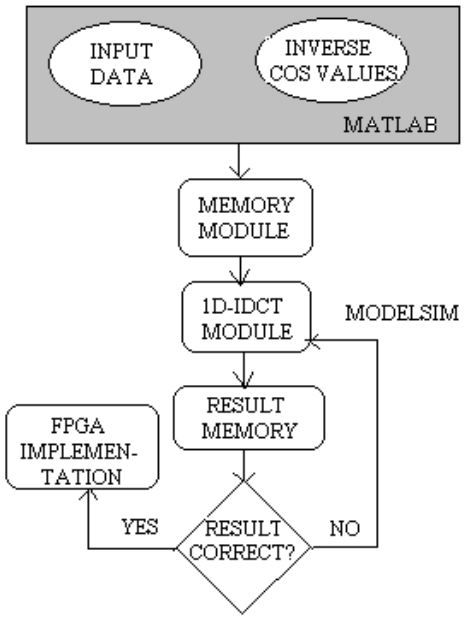

Figure 8. Implementation of 1D-IDCT

and the clock time period is 0.1µs. In the non-pipelined implementation, each set of output is obtained in 5 clock cycles, i.e. in 0.5µs, represented as t_n. In the pipelined implementation, each stage is registered, hence we get a set of eight outputs in each clock cycle, or 0.1µs, represented by t_p. The speed up of the pipelined implementation can be found using the formula:

$$S= (n*t_n)/ (k+n-1)*t_p \quad (4)$$

For example, if we set the number of data elements at 4000, then the speed comparison of both the implementations is given in the Table 1.

A comparison between a conventional and a five stage pipelined implementation of 1D-DCT/IDCT is given in Table II. The results have been obtained after implementing the design in Xilinx ISE 10.1 device XC3S4000. This comparison is based on the difference in the frequency of operation, number of registers, LUTs etc. Table II shows that the pipelined implementation uses fewer registers, LUTs, etc. This proves the design to be more efficient. The pipelined design is also more efficient with respect to power consumption. It needs to function only one-fifth of the time of a conventional design, or in other words, since the pipelined design needs to function for less time than a non-pipelined design, the power dissipation is also less.

TABLE I
COMPARISON OF SPEED OF PIPELINED AND NON-PIPELINED 1D DCT

Implementation	Non pipelined	Pipelined
Clock frequency	10MHz	10MHz
Number of data elements	4000	4000
Time consumed	2ms	0.4004ms
Speed up	1	4.995

TABLE II
COMPARISON OF PIPELINED AND NON-PIPELINED 1D DCT IMPLEMENTATION

Parameter	Non-pipelined	Pipelined
Operating frequency	82.65 MHz	83.996 MHz
Registers	2303	2228
Number of LUTs	2159	2094
Number of slices	2132	2059

VII. CONCLUSION

In this paper we have compared the results of implementing 1D DCT/IDCT both with and without pipelining. A speed up approximately equal to five is

obtained by pipelined implementation. The pipelined design also consumes less power, since it uses fewer multipliers and adders. The time taken to process a common set of data is $1/5^{th}$ of the time required by conventional implementations.

VIII. ACKNOWLEDGMENT

The authors gratefully acknowledge the Almighty who gave us strength and health to successfully complete this venture. We also wish to thank Amrita Vishwa Vidyapeetham, and in particular the Digital library, for granting us access to their research facilities and for providing us with the laboratory facilities necessary to conduct our research.

REFERENCES

[1] *D.W. Trainor J.P. Heron* and R.F. Woods," Implementation of the 2D DCT using a XILINX XC6264 FPGA", 0-7803-3806-5/97

[2] *Syed Ali Khayam*,"The Discrete Cosine Transform (DCT): Theory And Application1", Department of Electrical & Computer Engineering, Michigan State University

[3] *S. An C. Wang* "Recursive algorithm, architectures and FPGA Implementation of the two- dimensional discrete cosine transform", The Institution of Engineering and Technology 2008.

[4] *A.Aggoun and I. Jalloh*, "Two-dimensional DCT/SDCU architecture", 2003 IEE proceedings online no. *20030063*, ***DO:*** 10.1049/ip-dt:20030063 .

[5] *Khurram Bukhari, Georgi Kuzmanov and Stamatis Vassiliadis*" DCT and IDCT Implementations on Different FPGA Technologies", Computer Engineering Lab, Delft University of Technology.

[6] *Kuo-Hsing Cheng , Chih-Sheng Huang and Chun-Pin Lin* "The Design and implementation of DCT/IDCT Chip with Novel Architecture" May 28-31, 2000, Geneva, Switzerland.

[7] *Christoph Loeffler, Adriaan Lieenberg, and George S. Moschytz*, "Practical Fast 1-D Dct Algorithms with 11 Multiplications", cH2673- 2/89/0000-0098

[8] *Archana Chidanandan, Joseph Moder, Magdy Bayoumi*,"Implementation of NEDA-based DCT architecture using even-odd decomposition of the 8x 8 DCT matrix", 1-4244-0173-9/06.

[9] *Archana Chidanandan, Magdy Bayoumi*," **Area-Efficient Neda Architecture For The 1-D Dct/Idct",** 142440469X/06/.

[10] *S. Musupe and S. Arslun,* "Low power dct implementation approach for VLSI DSP processors", 0-7803-5471 -0/99 Ahmadreza NaghshNilchi, Third 2008 International Conference on Convergence and Hybrid Information Technology.