Design of High Throughput DCT Core Design by Efficient Computation Mechanism

Susrutha Babu Sukhavasi[1], Suparshya Babu Sukhavasi[1]
O. Ranga Rao[2], Sr. Sastry K.[2], G. Roopa Krishna Chandra[2]
[1]*Faculty, Department of ECE, K L University, Guntur, AP, India.*
[2] *Faculty, Department of ECE, Andhra Loyola College Of Eng &Tech ,Vijayawada, AP, India.*

Abstract : *This paper presents an approach towards VLSI implementation of the Discrete cosine Transform for image compression. Modules required for the corresponding ciruitThe design follows the JPEG standard and can be used for both lossy and lossless compression. In Discrete cosine transform, the filter implementation plays the key role. The poly phase structure is proposed for the filter implementation, which uses the Distributive Arithmetic (DA) technique. The implementation of DA based DCT IP core in ASIC (Application Specific Integrated Circuit). Which is popular in FPGA (Field Programmable Gate Array) implementations. To exploit the available resources on FPGAs, a new technique, which incorporates pipelining and parallel processing of the input samples, is proposed, with this architecture the speed (throughput) of the design increases. The intermediate coefficients obtained are efficiently stored in a memory and accessed for efficient memory utilization on FPGAs. The design is targeted on Xilinx spartan3, and the ASIC design of the same is carried out.*

Keywords: *DCT, Distributed Arithmetic.*

I. Introduction

Recently, multimedia applications are used widely in many embedded and portable systems, such as mobile phones, MP3 player and PDA, which require area efficient within high performance constraints. Therefore, area efficient design becomes a more important objective in Very Large Scale Integration (VLSI) designs Moreover; the multiplication unit always lies on the critical path and ultimately determines the performance and area efficient of arithmetic computing systems. To achieve high-performance and lengthen the battery lifetime, it is crucial to develop a compression with high-speed. In multimedia and digital signal processing (DSP) applications, many area efficient approaches have been presented. An efficient technique for calculation of sum of products or vector dot product or inner product or multiply and accumulate (MAC) MAC operation is very common in all Digital Signal Processing hardware designs. In a constant word length multiplier, the extra bits in the partial sums as well as the final product are truncated and no of binary adders required for accumulating 2 partial products equal to the wordlenth W. This causes truncation errors whose value not only depend on multiplicand and multiplier but also on arrangement of partial product accumulation. In general, the truncation part (TP) is usually truncated to reduce hardware costs in parallel shifting and addition operations, known as the direct truncation (Direct-T) method. Thus, a large truncation error occurs due to the neglecting of carry propagation from the TP to Main Part (MP). Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. It can also be extended to other sum functions such as complex multiplies, fourier transforms and so on. Look at my Radar on a chip paper for an application example of distributed arithmetic. In most of the multiply accumulate applications in signal processing, one of the multiplicands for each product is a constant. Usually each multiplication uses a different constant.Using our most compact multiplier, the scaling accumulator, we can construct a multiple product term parallel multiply-accumulate function in a relatively small space if we are willing to accept a serial input. In this case, we feed four parallel scaling accumulators with unique serialized data. Each multiplies that data by a possibly unique constant, and the resulting products are summed in an adder tree. If we stop to consider that the scaling accumulator multiplier is really just a sum of vectors, then it becomes obvious that we can rearrange the circuit. Here, the adder tree combines the 1 bit partial products before they are accumulated by the scaling accumulator. All we have done is rearranged the order in which the 1xN partial products are summed. Now instead of individually accumulating each partial product and then summing the results, we postpone the accumulate function until after we've summed all the 1xN partials at a particular bit time. This simple rearrangement of the order of the adds has effectively replaced N multiplies followed by an N input add with a series of N input adds followed by a multiply. This arithmetic manipulation directly eliminates N-1 Adders in an N product term multiply-accumulate function. For larger numbers of product terms, the savings becomes significant Further hardware savings are available when the coefficients C_n are constants. If that is true, then the adder tree shown above becomes a Boolean logic function of the 4 serial inputs. The combined 1xN products and adder tree is reduced to a four input look up table. The sixteen entries in the table are sums of the constant coefficients for all the possible serial input combinations. The table is made wide enough to accommodate the largest sum without overflow. Negative table values are sign extended to the width of the table, and the input to the scaling accumulator should be sign extended to maintain negative sums. Obviously the serial inputs limit the performance of such a circuit. As with most hardware applications, we can obtain more performance by using more hardware. In this case, more than one bit sum can be computed at a time by duplicating the LUT and adder tree as shown here. The second bit computed will have a different weight than the first, so some shifting is required before the bit sums are combined. In this 2 bit at a time implementation, the odd bits are fed to one LUT and adder tree, while the even bits are simultaneously fed to an identical tree. The odd bit partials are left shifted to properly weight the result and added to the even partials before accumulating the aggregate. Since two bits are taken at a time, the scaling accumulator has to shift the feedback by 2 places.This paralleling scheme can be extended to compute more than two bits at a time. In the extreme case, all input bits can be computed in parallel and then combined in a shifting adder tree. No scaling accumulator is needed in this case, since the output from the adder tree is the entire sum of products. This fully parallel implementation has a data rate that matches the serial clock, which can be greater than 100 MS/S in today's FPGAs. The Discrete cosine transform (DCT) is widely used in digital image processing, especially in image transform coding, as it performs much like the optimal Karhunen-Loeve transform (KLT) under a variety of criteria. Many algorithms for the computation of the DCT have been proposed since the introduction of the DCT by Ahmed, Natarajan, and Rao in 1974. However, though most of them are good software solutions to the realization of DCT, only a few of them are really suitable for VLSI implementation. Cyclic convolution plays an important role in digital signal processing due to its nature of easy implementation. Specifically, there exists a number of well-developed convolution algorithms and it can be easily realized through modular and structural hardware such as distributed arithmetic and systolic array . The way of data movement forms a significant part in the determination of the efficiency of the realization of a transform using the distributed arithmetic. The realization of a cyclic convolution with the distributed arithmetic requires only simple table look-up technique and some simple rotations of the corresponding data set. Hence, the cyclic convolution structure can be considered as the simplest form that is most suitable to be realized with the distributed arithmetic. It is because of this reason, one may consider that the basic criterion for the realization of a transform using the distributed arithmetic relies on the possibility of having an efficient way to convert the transform into the cyclic convolution form. If we could be able to convert a transform into the cyclic convolution form with the minimum number of operations, it would imply an optimal approach for the realization of the transform using the distributed arithmetic. Some basic formulations have been suggested for the realization of the DCT using the distributed arithmetic. In their formulations, they either still required some extra multiplications for their formulations or have to use cyclic convolutions of different lengths . The former case has the major problem that it violates the major advantage of the distributed arithmetic which replaces multiplications by additions. The latter case requires relatively complicated circuitry to allow the realization of cyclic convolutions of variable lengths. Different from the above approaches, one may also convert the DCT into the Discrete Fourier Transform (DFT) and make use of the famous algorithms to convert the corresponding DFT into cyclic convolution form. Indeed, this is a possible approach; however, it turns a real transform into a transform with complex numbers.The realization could still be complicated even if some simplification techniques are to be applied. The field of computer science is growing at the fastest pace since it started back in the early 20th century. We didn't have the actual computing machines until the mid of the 20th century but the algorithms for computing were already being developed from the beginning of the century. The advancement of the computing field has affected every imaginable field in the human lives. These fields include engineering, medicine, geology, meteorology, movies, and pictures etc. The high speed of digital computer has contributed to significant progress in the field of optics. Image processing by far has benefited significantly from the high speed digital computers Since the day the very first successful picture was taken in June of 1827, the quality and style of images have improved significantly. Niepce took this picture by using material that hardened on exposure to light . There were many different ways to taking these images before the camera that we know today was created. The main problem with all of the images taken was the fact that with every picture that was taken a lot of redundant and useless information was also stored. The problems range from faded, blurry and noisy pictures. All these useless data stored with the images is termed as Noise. Noise, in technical terms is defined as the irrelevant or meaningless data. Image restoration is one of the many branches of image processing. It is defined as a process that removes all the noise and irrelevant data from the images to make them clearer and better visible. The tools and techniques used in image processing are imported from signal processing. The main difference, however, between signal processing and image processing is that when binary data is compressed, it is essential that we get the same data back when it is decompressed. With the images, on the other hand, it is not required. When an image is decompressed it is in most cases enough to get a replica of the image as long as the mean square error is within certain set limits and tolerable. As mentioned above, there are a number of factors that can adversely affect the image quality. There have been a number of techniques introduced and being researched to enhance the image as well as to improve the usefulness of the data. Enhancement programs make the information more visible. One of the most popular enhancement techniques is the Histogram equalization. This technique redistributes the intensities of the image equally to the entire gray level range of the image. Convolution is an image processing technique which is essentially a sequence mask operating on pixel neighborhood. Convolution involves High Pass and Low pass filters. Other image processing techniques include noise filters, trend removing filters, edge detection, and image analysis tools etc. One of the important aspects of image processing is the reduction of the coded description of the image while keeping all the pertinent information. Data compression methods with zero information loss have been used on image data for some time. GIF, JPEG, MPEG etc are all examples of the tools used for data compression without the loss of the information. There are a various techniques used these days to capture and store the images in a compressed format to reduce their storage sizes and use a smaller space. There are basically two categories of compression techniques; lossless and lossy. The main difference between these two categories is that in lossy compression the quality of the image is reduced in order to achieve higher compression ratio. On the other hand, in a lossless compression the quality of the image is given a higher preference than the compression ratio. So it can be said that there is an apparent trade off between quality of the image and the compression ratio whenever we talk about the image compression. Lossy techniques include Transform Coding such as DCT, Wavelets and Gabor, Vector quantization, Segmentation and Approximation methods, Spline approximation methods i.e. Bilinear Interpolation/Regularization, Fractal coding which includes Texture synthesis, Iterative functions systems (IFS) and Recursive iterative functions systems (RIFS). Lossless compression techniques, on the other hand, includes Run length Encoding, Huffman Encoding, Entropy Coding (Lempel/Ziv) and Area coding . Discrete cosine transform (DCT) has become the most popular technique for image compression over the past several years. One of the major reasons for its popularity is its selection as the standard for JPEG. DCTs are most commonly used for non-analytical applications such as image processing and. signal-processing DSP applications such as video conferencing, fax systems, video disks, and HDTV. DCTs can be used on a matrix of practically any dimension. Mapping an image space into a frequency space is the most common use of DCTs. For example, video is usually processed for compression/decompression as 8 x 8 blocks of pixels. Large and small features in a video picture are represented by low and high frequencies. An advantage of the DCT process is that image features do not normally change quickly, so many DCT coefficients are either zero or very small and require less data during compression algorithms. DCTs are fast and, like FFTs, require calculation of coefficients. The entire standards employ block based DCT coding to give a higher compression ratio. Various different techniques and algorithms employed for DCT will be discussed in detail in section 3 where DCT will be explored. As the topic of my thesis suggests, I will be exploring the possibilities of discrete cosine transforms for multi-resolution analysis. The aim here is to see if we can get the same results in compression using DCT as we can get by using the wavelets. The word multi-resolution refers to the simultaneous presence of different resolutions. The notion of multi-resolution was introduced by Mallat and Meyer in the years 1988-89. Multi-resolution analysis provides a convenient framework for developing the analysis and synthesis filters . DCTs have been in used for compression for quite some time and have been very popular. As I mentioned earlier the main reason for DCTs popularity is the fact that it’s been used as a standard in JPEGs. Wavelets are considered better than DCT when it comes to getting better results in compression. The supporters of MPEGs and JPEGs claim that DCT provides very good results as far as the compression is concerned. If this claim is in fact true then the question is that why are so many people interested in Internet protocol streaming. The reason IP streaming is really good that there are algorithms that provide a lot more compression with the same video quality as MPEG-2 does. We can take a wavelet algorithm and keep the same video quality and use 1 Mbps for a very nice high quality movie that would take us 3 Mbps with normal MPEGs with DCT . The previous paragraph is good argument for choosing wavelets over DCT, however, the previous paragraph has been erroneous in the sense that researches use DCT and JPEG interchangeably. The arguments made in previous paragraph against DCT are in fact against the standard JPEG and not DCT. JPEG uses just a small part of DCT and should not be taken as standard DCT algorithms in comparisons with wavelet. The basic difference between DCT and wavelets is that in wavelets rather than creating 8 X 8 blocks to compress, wavelets decompose the original signal into sub-bands. Wavelets are basically an optimizing algorithm for 5 representing a lot of change in the pictures. With DCT algorithm, the 8 X 8 blocks can lose their crisp edges, whereas, with wavelets the edges are very well defined. I don’t really want to go in detail with the differences between wavelets and DCT but I just want to mention it to prove my case for the proposal of DCT for multi-resolution analysis. There is another compression method being developed call Fractals which is based on quadratic equations. This method is very well suitable with images which have patterns or a lot of repetitions. Now, if the wavelets produce much better results than DCT then why do we need to try DCT for multi-resolution? The reason is that there are certain drawbacks to wavelets specially in terms of computation time required. For the highest compression rates, it takes a longer time to encode. The other reason is that MPEG is already a standard using DCTs and computer hardware comes with MPEGs built in. There is hardly any hardware available in the market these days which comes with wavelets as a built in standard.

II. METHODOLOGIES

It is the process of analyzing the design of Discrete Cosine Transform (DCT) core with Error Compensated Adder Tree based on distributed arithmetic. An efficient technique for calculation of sum of products or vector dot product or inner product or multiply and accumulate (MAC) MAC operation is very common in all Digital Signal Processing hardware designs. An old technique that has been revived by the wide spread use of Field Programmable Gate Arrays (FPGAs) for Digital Signal Processing (DSP). DA efficiently implements the MAC using basic building blocks (Look Up Tables) in FPGAs .The “basic” DA technique is bit-serial in nature. DA is basically a bit-level rearrangement of the multiply and accumulate operation DA hides the explicit multiplications by without ROM look-ups an efficient technique to implement on Field Programmable Gate Arrays (FPGAs). The speed in the critical path is limited by the width of the carry propagation Speed can be improved upon by using techniques to limit the carry propagation. An increasing number of services and the growing popularity of high definition TV require higher coding efficiency. As the ongoing demand increases, for better compression performance of the latest video coding standard, the H.264/AVC (Advanced Video Coding) is formulated .The H.264/AVC is also known as MPEG-4 Part 10 (Wiegand,et al., 2003;Sullivan,et al.,2004; Richardson, 2003).An advantage of the H.264/AVC is the simplicity of its transform. Distributed Arithmetic (DA) is an effective method for computing inner products when one of the input vectors is fixed. It uses pre computed look-up tables and accumulators instead of multipliers for calculating inner products and has been widely used in many DSP applications such as DFT, DCT, convolution, and digital filters. In particular, there has been great interest in implementing DCT with parallel distributed arithmetic and in reducing the ROM size required in the implementations since the DA-based DCT architectures are known to have very regular structures suitable for VLSI implementations. Most DA-based DCT implementations use the original DCT algorithm, or the even-odd frequency decomposition of the DCT algorithm along with some memory reduction techniques such as the partial sum technique and/or the offset binary coding technique. On the other hand, the proposed architecture uses the DA-based DCT algorithm and requires less area than the conventional approaches, regardless of the memory reduction techniques employed in the ROM Accumulators (RACs).

MAIN MODULE'S:

- ERROR COMPENSATION CIRCUIT
- ERROR COMPENSATED ADDER TREE
- 1-D 8-POINT DCT DESIGN

III. MODULE DESCRIPTION:

Error Compensated Circuit:

Multipliers are commonly used components in digital signal processing applications (DSP). The multiplier produces 2n–bit output for n bit multiplicand and n bit multiplier input. But for some applications we may only require n bit multiplicand result. We can truncate n least-significant bits and preserve the m most significant bit. Although doing this would cause significant errors the area will be reduced half. To reduce truncation error we propose error compensation methods. In some applications these error could be ignored. The output will obtain MSBs using a rounding operation called post truncation (Post-T), which is used for high-accuracy applications. Hardware cost increases in the VLSI design. In general, the TP is usually truncated to reduce hardware costs in parallel shifting and addition operations, known as the direct truncation (Direct-T) method. Thus, a large truncation error occurs due to the neglecting of carry propagation from the TP to MP.

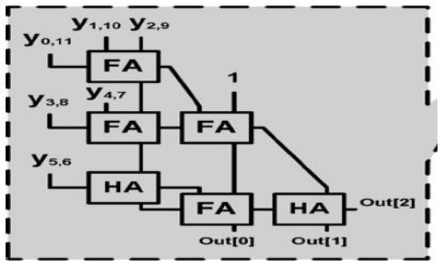


Fig 1: Error Compensation Circuit

In order to alleviate the truncation error effect, several error compensation bias methods have been presented.

All previous works were only applied in the design of a fixed-width multiplier. Because the products in a multiplier have a relationship between the input multiplier and multiplicand, the compensation methods usually use the correlation of inputs to calculate a fixed or an adaptive compensation bias using simulation or statistical analysis. The internal word-length usually uses 12 bits in a DCT design. Consequently, word length P=12 is chosen together with different Q values of 3, 6, 9 and12. The Post-T method provides the most accurate values for fixed-width computation nowadays. In addition, the Direct-T method has the largest inaccuracies of the errors for low-cost hardware design. The proposed ECAT is more accurate than Direct-T and is close to the performance of the Post-T method using a compensated circuit.

SUB MODULES'S

- HALF ADDER
- FULL ADDER
- ADDITION ELEMENTS

ERROR COMPENSATED ADDER TREE:

In this module consists of error compensated circuit, adder circuit, sign extension, zero extension. FA indicates a full-adder cell with three inputs (a, b, and c) and two outputs, a sum (s) and a carry-out (co). HA indicates half-adder cell with two inputs (a and b) and two outputs, a sum (s) and a carry-out (co).ECAT has the highest accuracy with a moderate area delay product. The shift-and-add method has the smallest area, but the overall computation time is longest. ECAT is suitable for high-speed and low-error applications. authorized doctors can login into the medical. Here also all the details about the doctor are registered by the medical admin. And the medical admin give the authentication details to the particular doctor after getting the authentication details doctor can login to the medical and can start the below processes.

SUBMODULES:

- ERROR COMPENSATED CIRCUIT
- PARTIAL PRODUCT GENERATION

ERROR COMPENSATED CIRCUIT

In this module the shifting and addition computation uses a shift-and-add operator in VLSI implementation in order to reduce hardware cost. However, when the number of the shifting and addition words increases, the computation time will also increase. Therefore, the shift-adder-tree (SAT) presented in operates shifting and addition in parallel by unrolling all the words needed to be computed for high-speed applications. The Q P-bit words operate the shifting and addition in par can be divided into two parts: the main part (MP) that includes P most significant bits (MSBs) and the truncation part (TP) that has least significant bits (LSBs).

PARTIAL PRODUCT GENERATION

In this module A truncated multiplier is an m × n multiplier with m bits output. Partial products can be divided into two subsets. The least significant part (LSP) includes the n less significant columns of the partial product matrix, while the most significant part (MSP) includes the remaining columns. The full-width multiplier output, P is given by P = SMSP + SLSP. Where SMSP and SLSP represent the weighted sum of the elements of MSP and LSP respectively. When a n bits output is needed, the most accurate choice is using the full rounded multiplier: it computes all the matrix of partial products, add a constant to the result on 2n bits and takes only the first n bits of the sum. The error introduced by the full rounded multiplier is calculated. Unfortunately the full rounded multiplier is the solution with the highest area occupation and power dissipation. A second possibility is using a truncated multiplier in which the partial products of the LSP are discarded assuming that their contribution to the n most significant bits of the output is negligible. This solution is very advantageous in terms of hardware performances.

IV. D 8 POINT DCT:

Image data compression has been an active research area for image processing over the last decade and has been used in a variety of applications. This paper investigates the implementation of an image data compression method with VLSI hardware that could be used in practical coding systems to compress JPEG signals. In practical situations, an image is originally defined over a large matrix of picture elements (pixels), with each pixel represented by a 8- or 16-bit gray scale value. This representation could be so large that it is difficult to store or transmit. The purpose of image compression is to reduce the size of the representation and, at the same time, to keep most of the information contained in the original image . DCT based coding and decoding systems play a dominant role in real-time applications. However, the DCT is computationally intensive. In addition, 2D-DCT has been recommended by standard organizations the Joint Photographic Expert Group (JPEG). The standards developed by these groups aid industry manufacturers in developing real-time 2D-DCT chips for use in various image transmission and storage systems.Generally there is a higher degree of correlation between the intensity values of adjacent pixels of an image. image compression can be achieved by removing such a redundant information. The discrete cosine transform (DCT) is widely used in most applications for compression of digital image and video signals. After performing the DCT operation on image or video signals most of energy is found to be concentrated in low frequency region. Recently hardware implementation of video coding standards fall into three main design categories :

1. Video signal processors,
2. Multimedia coprocessors,
3. Dedicated coders and decoders.

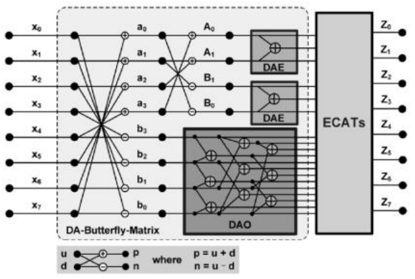


Fig 2: Architecture Of The Proposed 1-D 8-Point DCT

Video signal processors are programmable processors with a digital processing core or reduced instruction set computer core and co-processing unit for compute intensive operations such as motion estimation. Multimedia coprocessors are also programmable. The computation of DCT is on eof process of transforming a block of data from the spatial domain to frequency domain. Inverse process of restoring the spatial domain data from frequency domain is carried out through Inverse DCT (IDCT).The Discrete Cosine Transform (DCT) has gained the reputation of being a very effective signal analysis tool for many practical applications.

DISTRIBUTED ARITHMETIC

Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. It can also be extended to other sum functions such as complex multiplies, Fourier transforms and so on. Look at my Radar on a chip paper for an application example of distributed arithmetic. In most of the multiply accumulate applications in signal processing, one of the multiplicands for each product is a constant. Usually each multiplication uses a different constant. Using our most compact multiplier, the scaling accumulator, we can construct a multiple product term parallel multiply-accumulate function in a relatively small space if we are willing to accept a serial input. In this case, we feed four parallel scaling accumulators with unique serialized data.

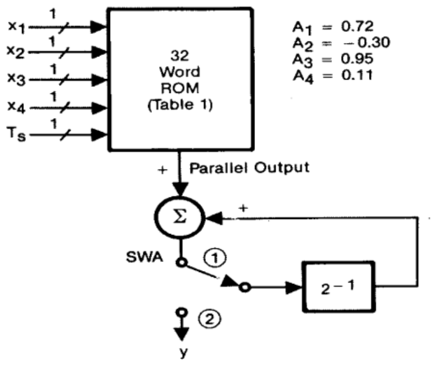


Fig 3: Distributed Arithmetic

Each multiplies that data by a possibly unique constant, and the resulting products are summed in an adder tree .If we stop to consider that the scaling accumulator multiplier is really just a sum of vectors, then it becomes obvious that we can rearrange the circuit. Here, the adder tree combines the 1 bit partial products before they are accumulated by the scaling accumulator. All we have done is rearranged the order in which the 1xN partial products are summed. Now instead of individually accumulating each partial product and then summing the results, we postpone the accumulate function until after we've summed all the 1xN partials at a particular bit time. This simple rearrangement of the order of the adds has effectively replaced N multiplies followed by an N input add with a series of N input adds followed by a multiply. This arithmetic manipulation directly eliminates N-1 Adders in an N product term multiply-accumulate function. For larger numbers of product terms, the savings becomes significant. Further hardware savings are available when the coefficients C_n are constants. If that is true, then the adder tree shown above becomes a Boolean logic function of the 4 serial inputs. The combined 1xN products and adder tree is reduced to a four input look up table. The sixteen entries in the table are sums of the constant coefficients for all the possible serial input combinations. The table is made wide enough to accommodate the largest sum without overflow. Negative table values are sign extended to the width of the table, and the input to the scaling accumulator should be sign extended to maintain negative sums. Obviously the serial inputs limit the performance of such a circuit. As with most hardware applications, we can obtain more performance by using more hardware. In this case, more than one bit sum can be computed at a time by duplicating the LUT and adder tree as shown here. The second bit computed will have a different weight than the first, so some shifting is required before the bit sums are combined.

DISTRIBUTED ARITHMETIC OPERATION

In this 2 bit at a time implementation, the odd bits are fed to one LUT and adder tree, while the even bits are simultaneously fed to an identical tree. The odd bit partials are left shifted to properly weight the result and added to the even partials before accumulating the aggregate. Since two bits are taken at a time, the scaling accumulator has to shift the feedback by 2 places This paralleling scheme can be extended to compute more than two bits at a time.

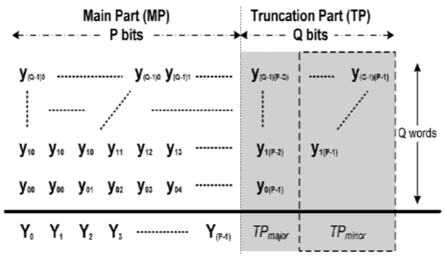

Main Part (MP)
P bits
Truncation Part (TP)
Q bits
Q words
$y_{(Q-1)0}$
$y_{(Q-1)0}$ $y_{(Q-1)1}$
$y_{(Q-1)(P-Q)}$
$y_{(Q-1)(P-1)}$
y_{10} y_{10} y_{10} y_{11} y_{12} y_{13}
$y_{1(P-2)}$ $y_{1(P-1)}$
y_{00} y_{00} y_{01} y_{02} y_{03} y_{04}
$y_{0(P-1)}$
Y_0 Y_1 Y_2 Y_3 $Y_{(P-1)}$
TP_{major}
TP_{minor}

FIG 4: Q , P-bit words shifting and addition operations in parallel

In the extreme case, all input bits can be computed in parallel and then combined in a shifting adder tree. No scaling accumulator is needed in this case, since the output from the adder tree is the entire sum of products. This fully parallel implementation has a data rate that matches the serial clock, which can be greater than 100 MS/S in today's FPGAs. Most often, we have more than 4 product terms to accumulate. Increasing the size of the LUT might look attractive until you consider that the LUT size grows exponentially. Considering the construction of the logic we stuffed into the LUT, it becomes obvious that we can combine the results from the LUTs in an adder tree. The area of the circuit grows by roughly 2n-1 using adder trees to expand it rather than the 2^n growth experienced by increasing LUT size. For FPGAs, the most efficient use of the logic occurs when we use the natural LUT size (usually a 4-LUT, although and 8-LUT would make sense if we were using an 8 input block RAM) for the LUTs and then add the outputs of the LUTs together in an adder tree. The Discrete cosine transform (DCT) is widely used in digital image processing, especially in image transform coding, as it performs much like the optimal Karhunen-Loeve transform (KLT) under a variety of criteria. Many algorithms for the computation of the DCT have been proposed since the introduction of the DCT by Ahmed, Natarajan, and Rao in 1974. However, though most of them are good software solutions to the realization of DCT, only a few of them are really suitable for VLSI implementation. Cyclic convolution plays an important role in digital signal processing due to its nature of easy implementation. Specifically, there exists a number of well-developed convolution algorithms and it can be easily realized through modular and structural hardware such as distributed arithmetic and systolic array . The way of data movement forms a significant part in the determination of the efficiency of the realization of a transform using the distributed arithmetic. The realization of a cyclic convolution with the distributed arithmetic requires only simple table look-up technique and some simple rotations of the corresponding data set. Hence, the cyclic convolution structure can be considered as the simplest form that is most suitable to be realized with the distributed arithmetic. It is because of this reason, one may consider that the basic criterion for the realization of a transform using the distributed arithmetic relies on the possibility of having an efficient way to convert the transform into the cyclic convolution form. If we could be able to convert a transform into the cyclic convolution form with the minimum number of operations, it would imply an optimal approach for the realization of the transform using the distributed arithmetic. Some basic formulations have been suggested for the realization of the DCT using the distributed arithmetic. In their formulations, they either still required some extra multiplications for their formulations or have to use cyclic convolutions of different lengths . The former case has the major problem that it violates the major advantage of the distributed arithmetic which replaces multiplications by additions. The latter case requires relatively complicated circuitry to allow the realization of cyclic convolutions of variable lengths. Different from the above approaches, one may also convert the DCT into the Discrete Fourier Transform (DFT) and make use of the famous algorithms to convert the corresponding DFT into cyclic convolution form. Indeed, this is a possible approach; however, it turns a real transform into a transform with complex numbers. The realization could still be complicated even if some simplification techniques are to be applied.

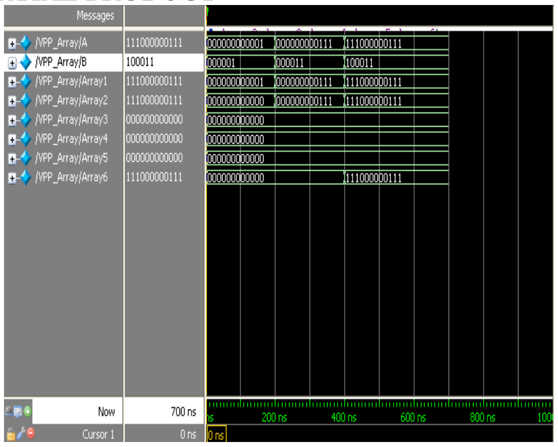
APPLICATIONS

The DCT core can be utilized for a variety of multimedia applications including:

- Office automation equipment (Multifunction printers, digital copiers etc)
- Digital cameras & camcorders
- Video production, video conference
- Surveillance systems

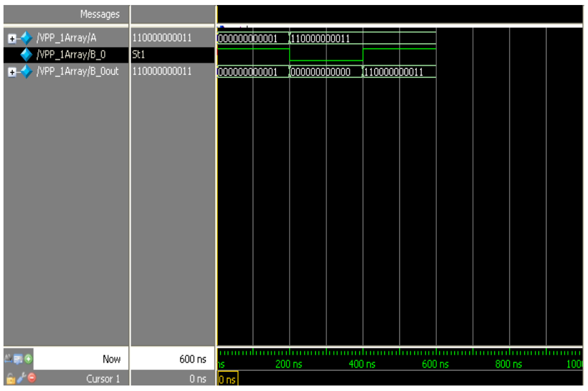
V. SIMULATION RESULTS

PARTIAL PRODUCT



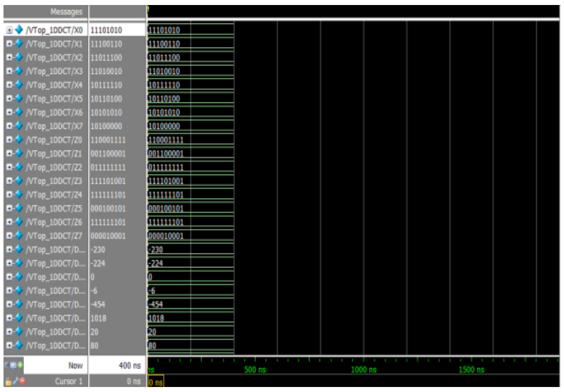
```
Messages
/VPP_Array/A        111000000111   000000000001 | 000000000111 | 111000000111
/VPP_Array/B        100011         000001       | 000011       | 100011
/VPP_Array/Array1   111000000111   000000000001 | 000000000111 | 111000000111
/VPP_Array/Array2   111000000111   000000000000 | 000000000111 | 111000000111
/VPP_Array/Array3   000000000000   000000000000
/VPP_Array/Array4   000000000000   000000000000
/VPP_Array/Array5   000000000000   000000000000
/VPP_Array/Array6   111000000111   000000000000                | 111000000111

Now       700 ns   ns  200 ns  400 ns  600 ns  800 ns  100
Cursor 1  0 ns     0 ns
```

PARTIAL PRODUCT1



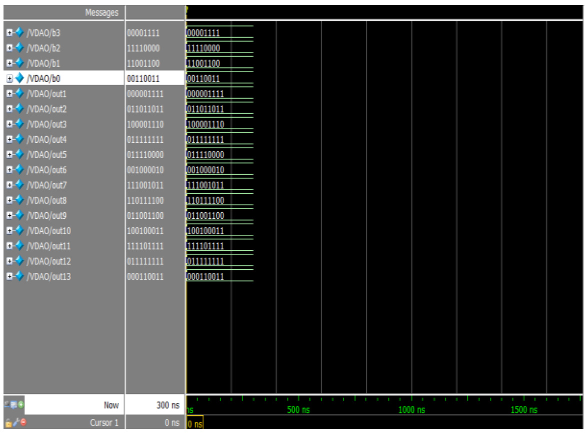
```
Messages
/VPP_1Array/A        110000000011   000000000001 | 110000000011
/VPP_1Array/B_0      St1
/VPP_1Array/B_0out   110000000011   000000000001 | 000000000000 | 110000000011
Now       600 ns
Cursor 1  0 ns
```

TOP MODULE



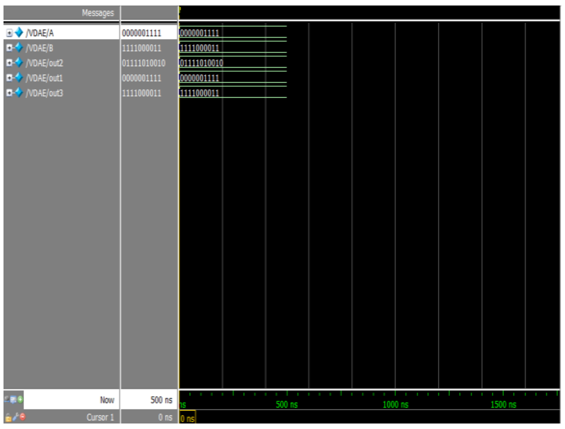
```
Messages
/VTop_1DDCT/X0    11101010
/VTop_1DDCT/X1    11100110
/VTop_1DDCT/X2    11011100
/VTop_1DDCT/X3    11010010
/VTop_1DDCT/X4    10111110
/VTop_1DDCT/X5    10110100
/VTop_1DDCT/X6    10101010
/VTop_1DDCT/X7    10100000
/VTop_1DDCT/Z0    110001111
/VTop_1DDCT/Z1    001100001
/VTop_1DDCT/Z2    011111111
/VTop_1DDCT/Z3    111101001
/VTop_1DDCT/Z4    111111101
/VTop_1DDCT/Z5    000100101
/VTop_1DDCT/Z6    111111101
/VTop_1DDCT/Z7    000010001
/VTop_1DDCT/D...  -230
/VTop_1DDCT/D...  -224
/VTop_1DDCT/D...  0
/VTop_1DDCT/D...  -6
/VTop_1DDCT/D...  -454
/VTop_1DDCT/D...  1018
/VTop_1DDCT/D...  20
/VTop_1DDCT/D...  80
Now       400 ns
Cursor 1  0 ns
```

DAO



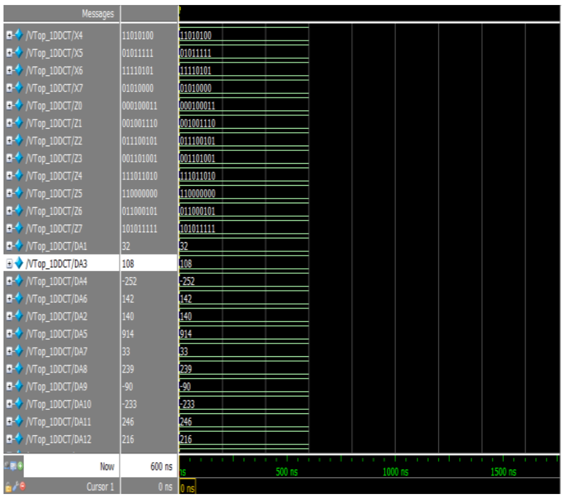
```
Messages
/VDAO/b3      00001111
/VDAO/b2      11110000
/VDAO/b1      11001100
/VDAO/b0      00110011
/VDAO/out1    000001111
/VDAO/out2    011011011
/VDAO/out3    100001110
/VDAO/out4    011111111
/VDAO/out5    011110000
/VDAO/out6    001000010
/VDAO/out7    111001011
/VDAO/out8    110111100
/VDAO/out9    011001100
/VDAO/out10   100100011
/VDAO/out11   111101111
/VDAO/out12   011111111
/VDAO/out13   000110011
Now       300 ns
Cursor 1  0 ns
```

DAE



```
Messages
/VDAE/A       0000001111
/VDAE/B       1111000011
/VDAE/out2    01111010010
/VDAE/out1    0000001111
/VDAE/out3    1111000011
Now       500 ns
Cursor 1  0 ns
```

BUTTERFLY MATRIX



```
Messages
/VTop_1DDCT/X4     11010100
/VTop_1DDCT/X5     01011111
/VTop_1DDCT/X6     11110101
/VTop_1DDCT/X7     01010000
/VTop_1DDCT/Z0     000100011
/VTop_1DDCT/Z1     001001110
/VTop_1DDCT/Z2     011100101
/VTop_1DDCT/Z3     001101001
/VTop_1DDCT/Z4     111011010
/VTop_1DDCT/Z5     110000000
/VTop_1DDCT/Z6     011000101
/VTop_1DDCT/Z7     101011111
/VTop_1DDCT/DA1    32
/VTop_1DDCT/DA3    108
/VTop_1DDCT/DA4    -252
/VTop_1DDCT/DA6    142
/VTop_1DDCT/DA2    140
/VTop_1DDCT/DA5    914
/VTop_1DDCT/DA7    33
/VTop_1DDCT/DA8    239
/VTop_1DDCT/DA9    -90
/VTop_1DDCT/DA10   -233
/VTop_1DDCT/DA11   246
/VTop_1DDCT/DA12   216
Now       600 ns
Cursor 1  0 ns
```

VI. CONCLUSIONS

This paper presents an FPGA implementation of efficient architecture for computing the 2-D DCT with distributed arithmetic. The proposed architecture requires less hardware than conventional architectures which use the original DCT algorithm or the even-odd frequency decomposition method. The modules of the transpose memory and parallel Distributed Arithmetic 2-D DCT architecture were designed and synthesized. The paper contributed with specific simplifications in the multiplier stage, by using shift and adds method, which lead to hardware simplification and speed up over architecture.

REFERENCES

[1]. A. M. Shams, a. Chidanandan, w. Pan, and m. A. Bayoumi, "NEDA: A Low-Power High-Performance DCT Architecture," IEEE trans. Signal process. vol. 54, no. 3, pp. 955–964, mar. 2006.

[2]. M. R. M. Rizk and m. Ammar, "Low Power Small Area High Performance 2d-Dct Architecture," in proc. Int. Design test workshop, 2007, pp. 120–125.

[3]. 3 y. Chen, x. Cao, q. Xie, and c. Peng, "An Area Efficient High Performance Dct Distributed Architecture For Video Compression," in Proc. Int. Conf. Adv. Comm. Technol., 2007, pp. 238–241.

[4]. C. Peng, X. Cao, D. Yu, and X. Zhang, "A 250 Mhz Optimized Distributed Architecture Of 2 D 8x8 DCT," in Proc. Int. Conf. ASIC, 2007, pp. 189–192.

[5]. C. Y. Huang, L. F. Chen, and Y. K. Lai, "A High-Speed 2-D Transforms Architecture With Unique Kernel For Multi-Standard Video Applications,"In Proc. IEEE Int. Symp. Circuits Syst., 2008, pp. 21–24.

[6]. S. S. Kidambi, F. E. Guibaly, and A. Antonious, "Area-Efficient Multipliersfor Digital Signal Processing Applications," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[7]. 7. A. Shams, A. Chidanandan, W. Pan, and M. Bayoumi, ―NEDA: A low power high throughput DCT architecture,‖ IEEE Transactions on Signal

[8]. Processing, vol.54(3), Mar. 2006.

[9]. 8. Peng Chungan, Cao Xixin, Yu Dunshan, Zhang Xing, ―A 250MHz optimized distributed architecture of 2D 8x8 DCT, 7Th InternationalConference on ASIC, pp. 189 – 192, Oct. 2007.

[10]. 9. B.G. Lee, ― A new algoritm to compute the discrete cosine transform―IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp. 1243-

[11]. 1245, Dect.1984.

[12]. 10. VijayaPrakash and K.S.Gurumurthy.‖A Novel VLSI Architecture for Digital Image Compression Using Discrete Cosine Transform and" Quantization IJCSNS September 2010.

[13]. 11. Nabeel Shirazi, Al Walters, and Peter Athanas ―Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines presented at the IEEE Symposium on FPGAs for Custom Machines, Napa Valley, California, April 1995.

Authors

S.Susrutha Babu was born in India, A.P. He received the B.Tech degree from JNTU, A.P, and M.Tech degree from SRM University, Chennai, Tamil Nadu, India in 2008 and 2010 respectively. He worked as **Assistant Professor** in Electronics Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in **K L University**. He is a member of Indian Society for Technical Education and International Association of Engineers. His research interests include antennas, FPGA Implementation, Low Power Design and wireless communications and Digital VLSI. He has published articles in various international journals and Conference in IEEE.

S.Suparshya Babu was born in India, A.P. He received the B.Tech degree from JNTU, A.P, and M.Tech degree from SRM University, Chennai, Tamil Nadu, India in 2008 and 2010 respectively. He worked as **Assistant Professor** in Electronics Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in **K L University**. He is a member of Indian Society for Technical Education and International Association of Engineers. His research interests include antennas, FPGA Implementation, Low Power Design and wireless communications and Robotics. He has published articles in various international journals and Conference in IEEE

S R Sastry Kalavakolanu was born in A.P,India. He received the B.Tech degree in Electronics & communications Engineering from Jawaharlal Nehru Technological University in 2010. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include Low Power VLSI Design. He has undergone 3 International Journals and 1 publishment in IEEE.

P.Bose Babu was born in A.P,India, He Completed M.Tech in VLSI system Design from MVGR COLLEGE OF ENGG.&TECH, Vizianagaram in 2011 and B.Tech from QIS College of engineering, in 2009 in Electronics &Communication engineering. Presently he is Working as Asst.Professor in ANDHRA LOYOLA college of Engg. & Technology, Vijayawada,A.P,India.

G.Roopa Krishna Chandra was born in A.P,India He received his B.Tech degree in Electronics and Communication Engineering from Sri Sarathi Institute of Engineering and Technology year 2009. He has completed M.Tech in Lakireddy Bali Reddy College of engineering in 2012. His Research areas are Adaptive Signal Processing, Embedded Systems.