IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735
PP 62-71
www.iosrjournals.org

Implementation of Next-Generation Video Codec In-Block Prediction-Based Scheme Using DTT

[1]S.Gomathi, PG Student, [2]MR. S.M. Balamurugan, Assistant Professor
1,2Department of EEE,GKM college of Engineering and Technology, Chennai

Abstract- *A low-complexity approximation for the discrete Tchebichef transforms (DTT). The proposed forward and inverse transforms are multiplication-free and require a reduced number of additions and bit-shifting operations. Numerical compression simulations demonstrate the efficiency of the proposed transform for image and video coding. Furthermore, Xilinx Virtex-6 FPGA based hardware realization shows 44.9% reduction in dynamic power consumption and 64.7% lower area when compared to the literature. The bandwidth reduction comes from two sources in our scheme, which differs from its previous designs and achieves a much higher compression ratio. First, it comes from pixel truncation. We use truncated pixels (PR) for integer motion estimation (IME) and acquire truncated residuals for factional motion estimation (FME) and motion compensation (MC). DCT algorithm has diverse applications and is widely used for Image compression. Here we propose discrete Tchebichef transform (DTT) which is a non trigonometric fully polynomial-based orthogonal transform. In terms of energy compaction and orthogonally DTT has similar properties as compared to DCT. Implementing DTT algorithm reduces the number of computations during processing, increases the accuracy of reconstruction of the image, and reduces the chip area of implementation of a processor built for this purpose. This reduces the overall power consumption.*
Index Terms*—Frame recompression, high-efficiency video coding (HEVC), motion estimation, video encoder approximate DTT, fast algorithms, image and video coding.*

I. INTRODUCTION

AS The next-generation standard of video coding, High Efficiency Video Coding (HEVC) [1] is intended to provide super high definition (HD), which offers significantly enhanced visual experience. Currently, 1080p HD has already become a mainstream standard for various video applications. Higher specifications such as 4K × 2 K quad full high definition (QFHD) format, which delivers at least four times the data throughput of HD, have been targeted by next generation applications. However, the corresponding huge external memory access challenges the design of real-time video encoder very large-scale integration.

The main source of external memory access comes from motion estimation (ME). For 1080p at 30 frames/s FHD video coding, the bandwidth requirement for integer motion estimation (IME) could be as high as 40 Gb/s, if it set the search range to be (−64, 63). For fractional motion estimation (FME) and motion compensation (MC), the bandwidth also can be as high as 3.1 and 1.6 Gb/s. For 4 K video coding with higher frame frequency, the bandwidth requirement could be more than ten times of FHD video coding. Huge external memory bandwidth dramatically increases the difficulty of integrated (IC) design, and also makes the input/output (IO) power consumption unacceptable.

There are three main approaches to reduce the external bandwidth of video encoder. First, fast ME algorithm reduces the search points in search range, such as 4SS [2] or TZ [3] algorithm can reduce at most more than 90% search points. As a fast algorithm makes pixel data access random and unpredictable, it is more suitable for CPU- or DSP-based soft video encoder. For hardware encoder, some other approaches are proposed, such as pixel truncation [17] and hierarchical ME [9]. Secondly, many on-chip data reuse methods are proposed for ME. Tuan *et al.* [4] summarized data-reuse schemes as four levels, levels A–D. The main idea of data reuse is to reduce external memory access through increasing on-chip memory buffer. For example, the level D data reuse needs to store several lines of pixel in frame width on chip, and it can eliminate all of redundant memory access. However, it is too expensive to store lines of pixel on chip for FHD or 4 K video. The third approach is frame recompression. It is a technique to reduce the reference frame size by compressing the original pixel data before such data are stored into offchip memory. With a decreased amount of fetched data, the bandwidth requirement is reduced as a consequence. The compression algorithm benefits from its fast, high compression efficiency, and ultralow loss or lossless quality.

The frame recompression techniques have been widely studied in the past decade. The transform-based approach is proposed in [5]. However, this approach requires a large amount of computations which is not suitable for low-latency and low-power consumption systems. A down sampling-based recompression method is proposed in [6]. However, the PSNR may be degraded owing to lossy reference frame. In addition, the differential pulse code modulation (DPCM) is a widely used spatial domain compression method [7]–[10]. It achieves high compression efficiency by reducing the spatial redundancies of image. Many domain-specific methods are also proposed, for example, a lossless frame recompression method for hierarchical ME scheme is presented in [9], but this method can only be applied to some special ME fast search algorithms. TZ search and some other popular fast algorithms cannot be supported by this scheme.

The rest of this paper is organized as follows. In Section II, frame recompression is briefly introduced. The data dependency and current solutions are presented. Section III describes the proposed frame recompression scheme. In Section IV, the compression ratio of proposed scheme is evaluated and compared with other methods. Section V gives the implementation of video encoder with proposed mixed lossy and lossless (MLL) scheme, and evaluates the video quality under multibit TBT. Finally, the conclusion is presented in Section VI.

II. IMAGE PROCESSING SYSTEM AND MOTIVATION

A. Introduction

In most of the existing frame recompression methods are designed independently of video encoder. Unfortunately, they did not intend to finely cooperate with encoder, but only focus on improving the compression ratio. Actually, many information can be used for frame recompression and make it work more efficiently with encoder. For example, IME, FME, and MC have different workload and bandwidth requirement, the precision of compression can also be different for each of them. What is more, intra prediction in video encoder can be used to guide the recompression of each coding blocks. Frame recompression should take all of them into consideration, and make use of information from encoder. In this paper, an encoder friendly frame recompression scheme is proposed. This scheme fully uses the information from encoder and discriminate IME from FME and MC to finely save external memory bandwidth. There are two coding layer in our scheme: base layer (BL) and enhancement layer (EL). Base layer is lossy data which only used for IME. Combining with enhancement layer, lossless data can be reconstructed and it is used for FME and MC. Base layer is compressed by proposed three new techniques: tailing-bit truncation (TBT), in-block prediction (IBP), and small-value optimized variable length coding (SVO-VLC).

Multimedia data processing, which encompasses almost every aspects of our daily life such as communication broad casting, data search, advertisement, video games, etc has become an integral part of our life style. The most significant part of multimedia systems is application involving image or video, which require computationally intensive data processing. Moreover, as the use of mobile device increases exponentially, there is a growing demand for multimedia application to run on these portable devices. A typical image/video transmission system is shown in the Figure 1.1.

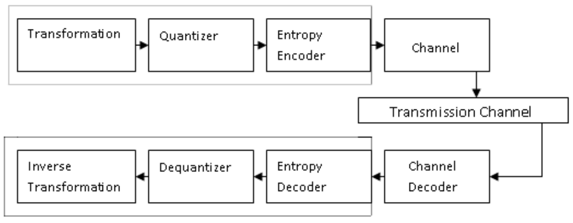


Figure1.1. Image/Video Transmission System

Multimedia files are large and consume lots of hard disk space. The files size makes it time-consuming to move them from place to place over school networks or to distribute over the Internet. Compression shrinks files, making them smaller and more practical to store and share. Compression works by removing repetitious or redundant information, effectively summarizing the contents of a file in a way that preserves as much of the original meaning as possible.

In order to reduce the volume of multimedia data over wireless channel compression techniques are widely used. Efficacy of a transformation scheme can be directly gauged by its ability to pack input data into as few coefficients as possible. This allows the quantizer to discard coefficients with relatively small amplitudes without introducing visual distortion in the reconstructed image.

B. Discrete cosine transform

Discrete cosine transform (DCT) is one of the major compression schemes owing to its near optimal performance and has energy compaction efficiency greater than any other transform. The principle advantage of image transformation is the removal of redundancy between neighbouring pixels. This leads to uncorrelated transform coefficients which can be encoded independently. DCT has that de correlation property.

The transformation algorithm needs to be of low complexity. Since the DCT is separable 2-D can be obtained from two 1-D DCTs. The 2-D DCT equation is given by Equation (1)

$$C(u,v)=\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right], \qquad 1$$

For u,v= 0,1,2,…,$N-1$.

The inverse transform is defined by Equation (2)

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right],$$

For x,y= 0,1,2,…,$N-1$. The 2-D basis functions can be generated by multiplying the horizontally oriented 1-D basis functions with vertically oriented set of the same functions.

Real value	Binary number	CSD number
0.4904	0011 1111	0100 000$\bar{1}$
0.4619	0011 1011	0100 $\bar{1}$011
0.4157	0011 0101	0011 0101
0.3536	0010 1101	0010 1101
0.2778	0010 0100	0010 0100
0.1913	0001 1000	0001 1000
0.0975	0000 1100	0001 0$\bar{1}$00

Fig 1.2 DCT co-efficients

In image compression, the image data is divided up into 8x8 blocks of pixels. (From this point on, each color component is processed independently, so a "pixel" means a single value, even in a color image.) A DCT is applied to each 8x8 block. DCT converts the spatial image representation into a frequency map: the low-order or "DC" term represents the average value in the block, while successive higher-order ("AC") terms represent the strength of more and more rapid changes across the width or height of the block. The highest AC term represents the strength of a cosine wave alternating from maximum to minimum at adjacent pixels.

C. Quantization

To discard an appropriate amount of information, the compressor divides each DCT output value by a quantization coefficient and rounds the result to an integer. The larger the quantization coefficient, the more data is lost, because the actual DCT value is represented less and less accurately. Each of the 64 positions of the DCT output block has its own quantization coefficient, with the higher-order terms being quantized more heavily than the low-order terms i.e. the higher-order terms have larger quantization coefficients. The resulting coefficients contain a significant amount of redundant data. Huffman compression will losslessly remove the redundancies, resulting in smaller data.

The human eye is able to distinguish between small differences in brightness over a relatively large area But it is not capable of distinguishing the exact strength of a high frequency brightness variation so good. Hence we can greatly reduce the amount of information in the high frequency components. This is achieved by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This rounding operation is the only lossy operation in the whole process if the DCT computation is performed with sufficiently high precision. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to represent.

A typical quantization matrix, as specified in the original JPEG Standard, is as follows:

$$Q=\begin{bmatrix}16 & 11 & 10 & 16 & 24 & 40 & 51 & 61\\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55\\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56\\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62\\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77\\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92\\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101\\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99\end{bmatrix}.$$

D. Entropy Coding

Entropy coding is a form of lossless data compression. The steps involved in entropy encoding are arrangement of image components in a zigzag manner employing run length encoding (RLE) algorithm. The

RLE groups similar frequencies together, inserting length coding zeros, and then using Huffman encoding on what is left. The arithmetic coding technique is mathematically superior to Huffman coding and the JPEG standard also allows its though it is not mandatory. Arithmetic coding typically makes files about 5–7% smaller. The previous quantized DC coefficient is used to predict the current quantized DC coefficient. The difference between the two is encoded rather than the actual value. The encoding of the 63 quantized AC coefficients does not use such prediction differencing.

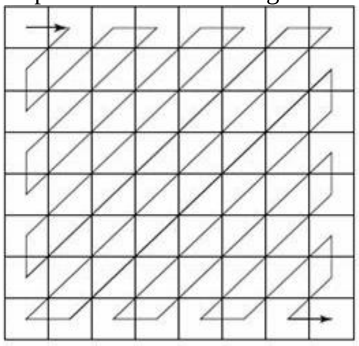


Figure 1.3 Zigzag ordering

The transformation unit plays a vital role in image compression as it concentrates the information in the low frequency ranges allowing the high frequency with less or nil information being discarded. Many DCT architectures based on different algorithms have been proposed.

E. Motivation

Most of current pixel compression schemes only achieve about 50% compression ratio, and it seems hard to go a short step. Because the improvement from spatial redundancy of original pixel is very hard, we decided to gain more from truncated pixel. Pixel truncation has already been used in motion estimation to reduce memory bandwidth [17], and truncated pixel will greatly remove the variations between neighbouring points. From our experiments, it can achieve 15.4% compression ratio after 3-b truncation. On combining pixel truncation and compression, much higher bandwidth reduction than current designs could be can achieved. Especially for video encoder, the truncated pixels can be only used for motion estimation, and then padding the residuals to reconstruct lossless pixels for motion compensation. It causes negligible video quality drop.

III. PROPOSED SYSTEM AND DTT SCHEME

A. Introduction

DCT algorithm has diverse applications and is widely used for Image compression. Here we propose discrete Tchebichef transform (DTT) which is a non trigonometric fully polynomial-based orthogonal transform.

In terms of energy compaction and orthogonally DTT has similar properties as compared to DCT. Implementing DTT algorithm reduces the number of computations during processing, increases the accuracy of reconstruction of the image, and reduces the chip area of implementation of a processor built for this purpose. This reduces the overall power consumption.

B. Hardware Complexity

The Complexity DTT implementation is also reduced by using only binary shift and addition operations. Moreover, the computational accuracy can be selected based on the trade-off between the hardware Complexity and approximation error. In addition, since our proposed algorithm has uniform post-scaling factor, it is also suitable for scaled DTT implementation.

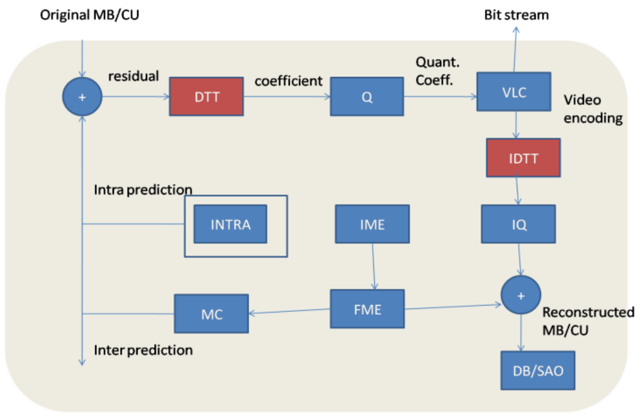


Fig1.4. Proposed system using DTT implementation

C. Advantages

Number of gates required in hardware implementation, such as on an FPGA, is minimum as hardware complexity is greatly reduced compared to other processors such as DSP multipliers and It is relatively simple in design. No multiplication and only addition, subtraction and bit-shifting operation ensures simple VLSI implementation.

D.DTT Scheme

The discrete Tchebichef transform (DTT) is a useful tool for signal coding and data decorrelation [1]. In recent years, signal processing literature has employed the DTT in several image processing problems, such as artefact measurement [2], blind integrity verification [3], and image compression [4]–[7]. In particular, the 8-point DTT has been considered in blind forensics for integrity check of medical images [3]. For image compression, the 8-point DTT is also capable of outperforming the 8-point discrete cosine transform (DCT) in terms of average bit-length in bitstream codification [4]. Moreover, in [7] an 8-point DTT-based encoder capable of improved image quality and reduced encoding/decoding time was proposed; being a competitor to state-of-the-art DCT-based methods. However, to the best of our knowledge, literature archives only one fast algorithm for the 8-point DTT, which requires a significant number of arithmetic operations [6]. Such high arithmetic complexity may be a hindrance for the adoption of the DTT in contemporary devices that demand low-complexity circuitry and low power consumption [8]–[10].

An alternative to the exact transform computation is the employment of approximate transforms. Such approach has been successfully applied to the exact DCT, resulting in several approximations [11], [12]. In general, an approximate transform consists of a low-complexity matrix with elements defined over a set of small integers, such as . The resulting matrix possesses null multiplicative complexity, because the involved arithmetic operations can be implemented exclusively by means of a reduced number of additions and bit-shifts. Prominent examples of approximate transforms include: the signed DCT [13], the series of DCT approximations by Bouguezel-Ahmed-Swamy [14]–[16], the approximation by Lengwehasatit-Ortega [17], and the integer based approximations described in [11], [12], [18], [19]. In this work, we introduce a low-complexity DTT approximation that requires 54.5% less additions than the exact DTT fast algorithm. The proposed method is suitable for image and video coding, capable of processing data coded according to popular standards—such as JPEG [20], H.264 [21], and HEVC [22]—at a low computational cost. Moreover, the FPGA hardware realization of the proposed transform is also sought. This letter unfolds as follows. Section II describes the DTT and introduces the approximate DTT with its associate fast algorithm. A computational complexity analysis is offered. In Section III, we perform numerical experiments; applying of the proposed transform as a tool for image and video compression. In Section IV, we provide very large scale integration (VLSI) realizations of the exact DTT and proposed approximation.

IV. EVALUTION OF BLOCK PREDICTION SCHEME

A. Tail Bit Truncation

Tail Bit Truncation (TBT) is a right-shifting operation to remove the tail bit of original pixels. Take 1-b TBT as an example (Fig. 4), 8 × 8 original pixels with 8-b sampling are loaded into TBT encoder. For each pixel, the 8-b original data is right shifted 1 b. As a result, two data sets are generated by TBT: truncated pixels and TR. The bit0 of all pixels are assembled together as TR, which will be stored to external memory directly. The remained 7-b data (bit7–bit1) comprise the truncated pixel, which will be output to IBP for further compression. There are two kinds of TBT decoding: partial decoding and full decoding. The truncated 7-b pixels are generated by SVO-VLC decoding and IBP. Partial decoding only left-shift the truncated pixel to 8-b, and add zero to least significant bit (LSB). 1-b lossy pixels can be reconstructed in partial decoding. Full decoding adds one more operation after partial decoding, it replaces the LSB zeros by TR. And the original lossless pixels are reconstructed.

The benefit of TBT is that it almost has no effect to video quality when doing IME under 1-b precision loss of pixel. The pixel value after truncation ranges from 1 to 127. The smaller range of pixel data makes the residual of IBP smaller and produces more zeros than nonzeros, which benefits the compression efficiency of SVO-VLC.

Moreover, TBT can be extended to multibit truncation. For multibit truncation, the compression ratio of truncated pixels is higher than 1-b truncation. However, the video quality will be affected since IME search has lower precision. The detailed discussion of multibit truncation is presented in Section V.

B. Intra Mode Referenced In-Block Prediction

IBP is used to compress the truncated pixels from TBT. To achieve high compression ratio, many prediction methods and modes are used in our scheme. As shown in Fig. 5, the inputs of IBP are 8 × 8 truncated block. The input pixels are categorized into three types: initial pixel (IP) which located in (0, 0), basic pixels (BP) which located in top and left, and remained are normal pixels (NP). Two main procedures are used in IBP: P1 prediction and P2 prediction. For P1 prediction, it uses IP and BP to do vertical and horizontal DPCM prediction, and generate BP residual. IP is the start point of P1 prediction, as shown in the right part of Fig. 5. The detailed function of P1 is shown as below.

P1 Prediction:

1) Vertical Prediction

$$P_{\text{pred}}(0, x) = BP(0, x-1) \tag{3}$$

$$BP_{\text{Residual}}(0, x) = BP(0, x) - P_{\text{pred}}(0, x). \tag{4}$$

2) Horizontal Prediction

$$P_{\text{pred}}(x, 0) = BP(x-1, 0) \tag{5}$$

$$BP_{\text{Residual}}(x, 0) = BP(x\ 0) - P_{\text{pred}}(x, 0). \tag{6}$$

For P2 prediction, it uses both of IP and BP as reference pixels. Intra mode from encoder is used to reduce mode prediction computation. Three candidate modes are selected according to intra mode, and one best NP prediction is chosen after mode decision. As shown in top right part of Fig. 5, every pixel has four reference pixels for prediction. C is the pixel to be predicted, R1 ∼ R4 are neighboring reference pixels. The P2 prediction can be described as

P1 Prediction:

$$^{C}(x,y) = {}^{f\,(\,R1}(x-1,y)\ ^{R2}(x-1,y-1)^{,}\ ^{R3}(x,y-1)\ ^{R4}(x+1,y-1)^{)} \tag{7}$$

$$NP_{\text{residual}}(x, y) = NP(x, y) - C_{(x,y)} \tag{8}$$

f () is a prediction function based on IBP mode, and its detailed definition is listed in Table I. To gain a good pre-diction result, totally eight prediction modes are proposed in this paper. More IPB modes produce more accurate pixels. However, it also brings more mode bits and increases the complexity of mode decision. Considering the massive computation of mode decision, the intra mode from video encoder is used to do premode decision. Table II shows a mode mapping from HEVC intra modes. There are 35 modes in the HEVC intra prediction. Each intra prediction mode maps to three IBP candidate modes. Because intra prediction is always performed in video encoder, every 8 × 8 coding block can guarantee an intra prediction mode from encoder. Another effort to reduce IBP complexity is that it only uses addition and shifting, which introduce very little computational overhead in both of CPU and digital circuit.

The output of IBP includes IP, IBP mode, BP residual and NP residual. VLC coding will be performed to reduce size of data. For MLL encoding path, all of operations mentioned above should be performed, whereas for MLL decoding path, it becomes much simpler as it does not needed to do mode decision.

C. Small-Value Optimized Variable Length Coding

Most of the residuals from IBP are small value. As we adopt TBT in MLL scheme, it produces smaller value than other schemes. Smaller value gives the chance to get smaller data after compression. To find a suitable VLC coding method, the residual data is analyzed as shown in Fig. 6. The results come from seven FHD video sequences.

D. External Memory Organization

The external memory organizations are two memory banks for reference pixel storage: PR bank and TR bank, and one memory bank for address table (TLB) storage. Two types of addressing method are used in our design: TLB addressing and fixed addressing. For fixed length data, such as TR and TLB data, it can be addressed by their base address and block index. For PR data, the data size is variable. To continuously load and store data, PR data is byte aligned, and the TLB is used to record start address and length of each 8 × 8 block.

For FHD video sequence, there are totally 32 640 8 × 8 partitions. It is not a good idea to store all of these TLBs on-chip. A small on-chip TLB SRAM with a separate TLB memory bank is required. Before fetching PR data, the corresponding TLB should be loaded in advance. Even it causes overhead, considering PR data is much more than TLB data, this cost is acceptable. Furthermore, the search window (SW) can be loaded row by row, and it can only preload the leading and ending block's TLB to further reduce overhead. To improve the external memory efficiency, TLB data is better to be accessed continuously. For example, SW can be loaded row by row, only TLBs in vertical boundary are needed. In this way, the TLB data is stored column by column in TLB memory bank. A typical TLB data for FHD video encoder is provided.

A simple MMU is used to handle all of memory addressing and TLB refreshing. The 8 × 8 block entry address for luma and chroma can be generated based on frame number, PR/TR base address, luma/chroma selector, block coordinate, and on-chip TLB SRAM.

V. IMPLEMENTATIONS AND RESULTS

A. Overview

A primary objective of this project was to develop a synthesizable model for the decoding algorithm. Synthesis is the process of converting the register transfer level (RTL) representation of a design into an optimized gate-level netlist. This is a major step in ASIC design flow that takes an RTL model closer to a low-level hardware implementation. Synthesis consists of three main steps. The first step is the *"Translation",* which involves converting the RTL description of a design into a non-optimized intermediate representation that is used by the synthesis tool. The second step is the *"logic optimization"*, which optimizes the internal representation by removing redundant logic and performing Boolean logic optimizations. The third step is called *"technology mapping & optimization"* which maps the internal representation to an optimized gate level representation using the technology library cells based on design constraints. In this chapter, we describe how the QUARTUS II tool was utilized to synthesize the verified Verilog model, by using a script that was developed to perform the synthesis based on certain constraints. The script generates several reports about the synthesis outcome including timing and area estimates.

B.Sythesis Methodology

The first step in the synthesis process is to read all the components in the design hierarchy. There are three components in the 3-level design hierarchy that needs to be synthesized. Since the RTL model utilizes a Verilog *"Package"*, then the synthesis tool needs to enable the semantics of a package. In addition, the synthesis tool needs to know if there are multiple instances of calling an automatic function in the design, to preserve separate values for each instance. After this step, a 50MHz clock signal is applied to the clock port of the root module, and the synthesis tool is programmed not to modify the clock tree during the optimization phase. In addition, an arbitrary input delay of 5ns with respect to the clock port is applied to all input and output ports (except the clock port itself) to set a safe margin by considering any unintended source of delay such as the delay associated with driving module/modules.

Finally, the tool compiles the design with high effort and reports any warning related the mapping and final optimization step. At the end, the tool generates reports for the optimized gate level netlist area, the worst combinational path timing, and any violated design constraint.

C. Synthesis Schematic Report

The synthesis tool optimizes the combinational paths in a design. In General, four types of combinational paths can exist in any design: [1] Input port of the design under test to input of one internal flip-

flip [2] Output of an internal flip-flop to input of another flip-flip [3] Output of an internal flip-flop to output port of the design under test [4] A combinational path connecting the input and output ports of the design under test.

The last DC command in the script developed in previous section, instructs the tool to report the path with the worst timing. In this case, the path with the worst timing is a combinational path of type two.

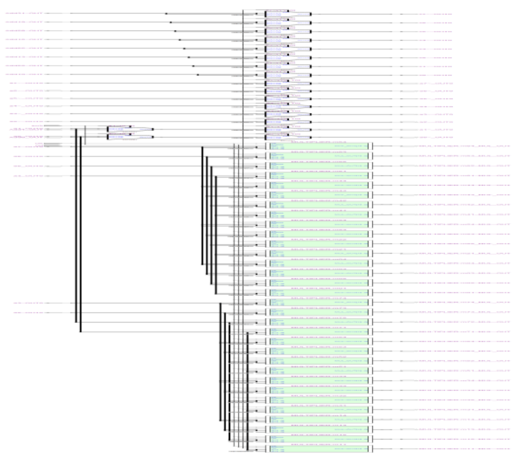


Fig 1.5 RTL Schematic Reports

D. MATLAB result

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. For data analysis, such as image to spatial data sequence conversion, image reconstruction from pixel values, PSNR calculations are done using MATLAB.

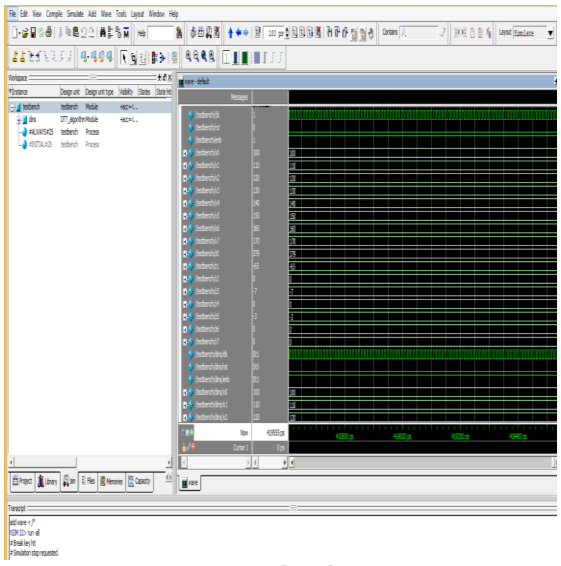
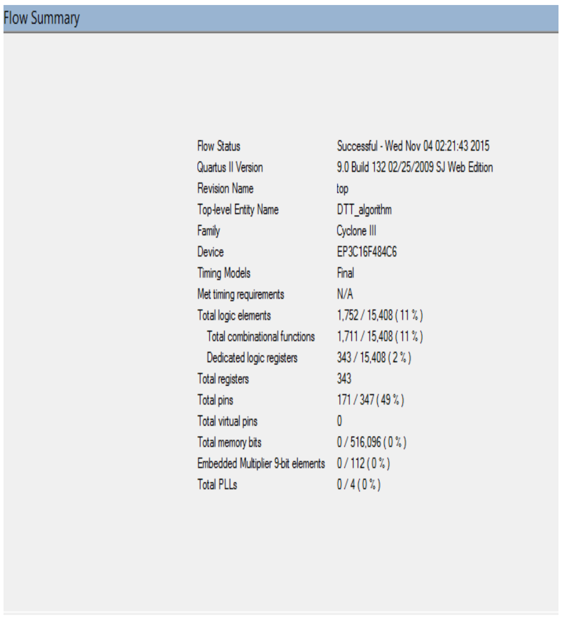


Fig1.5 *simulated output*

E. Synthesis Area Result

The synthesis area report shows the total number of cells and nets in the netlist. It also uses the area parameter associated with each cell in the LSI_10K library file, to calculate the total combinational and sequential area of the netlist. The total area of the gate level netlist is unknown since it depends on total area of the interconnects, which itself is a function of the wiring load model used in physical design. The total cell area in the netlist is reported as 22978 units, which is the sum of combinational and sequential areas. The synthesis area report is shown below:

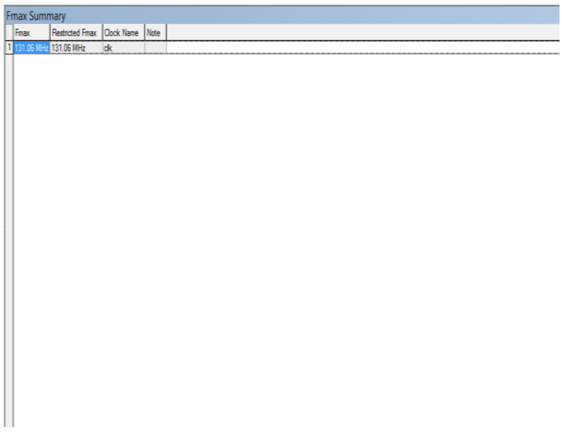


Flow Summary

Flow Status	Successful - Wed Nov 04 02:21:43 2015
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	top
Top-level Entity Name	DTT_algorithm
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Met timing requirements	N/A
Total logic elements	1,752 / 15,408 (11 %)
Total combinational functions	1,711 / 15,408 (11 %)
Dedicated logic registers	343 / 15,408 (2 %)
Total registers	343
Total pins	171 / 347 (49 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Fig 1.6 Area Utilisation

F. Performance Report for DTT



Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	131.06 MHz	131.06 MHz	clk	

Fig 1.7 Fmax summary report of slow corner.

G. Summary

The design is scripted as a Verilog file and synthesized using Quartus II 9.0 v. The design is synthesized into Cyclone device. The image is converted into pixels using MATLAB and the values are stored as a text file. The text file is accessed by the Model Sim ALTERA and the corresponding 2-D DCT coefficients are calculated. These values are then fed to the IDCT module which returns the spatial data sequence. These data are written to a text file. The image can be reconstructed from the text file using MATLAB coding. The simulated results are shown below.

VI. CONCLUSION AND FUTURE WORK

In this paper, the performance of DCT and DTT algorithm is analyzed for prediction based video compression system. Initially the implementation of DCT and its funtionlaity using MODELSIM is done.The proposed algorithm is based on MC to finely save external memory bandwidth. Then, the performance of DTT algorithm in numerical simulations is illustrated and this algorithm shows a significant performance improvement when compared to DCT, while the complexity is much lower compared to DCT based approach. The future work is to extend the objective of complexity reduction with hierarchical based task mapping area complexity will be reduced. To optmize the memory reduction by adding memory scheduling overall memory bits usage will be reduced.

REFERENCES

[1]. R. Mukundan, S. Ong, and P. A. Lee, “Image analysis by Tchebichef moments,” *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1357–1364, 2001.

[2]. L. Leida, Z. Hancheng, Y. Gaobo, and Q. Jiansheng, “Referenceless measure of blocking artifacts by tchebichef kernel analysis,” *IEEE Signal Process. Lett.*, vol. 21, pp. 122–125, Jan. 2014.

[3]. H. Huang, G. Coatrieux, H. Shu, L. Luo, and C. Roux, “Blind integrity verification of medical images,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, pp. 1122–1126, Nov. 2012.

[4]. F. Ernawan, N. Abu, and N. Suryana, “TMT quantization table generation based on psychovisual threshold for image compression,” in *2013 Int. Conf. Information and Communication Technology (ICoICT)*, Mar. 2013, pp. 202–207.

[5]. S. Prattipati, M. Swamy, and P. Meher, “A variable quantization technique for image compression using integer tchebichef transform,” in *2013 9th Int. Conf. Information,Communications and Signal Processing (ICICS)*, Dec. 2013, pp. 1–5.

[6]. S. Prattipati, S. Ishwar, P. Meher, and M. Swamy, “A fast integer tchebichef transform and comparison with integer cosine transform for image compression,” in *2013 IEEE 56th Int. Midwest Symp. Circuits and Systems (MWSCAS)*, 2013, pp. 1294–1297.

[7]. R. Senapati, U. Pati, and K. Mahapatra, “Reduced memory, low complexity embedded image compression algorithm using hierarchical listless discrete tchebichef transform,” *IET Image Process.*, vol. 8, pp. 213–238, Apr. 2014.

[8]. L. W. Chew, L.-M. Ang, and K. P. Seng, “Survey of image compression algorithms in wireless sensor networks,” in *2008 Int. Symp. Information Technology (ITSim)*, Aug. 2008, vol. 4, pp. 1–9.

[9]. F. Ernawan, E. Noersasongko, and N. Abu, “An efficient tchebichef moments for mobile image compression,” in *2011 Int. Symp. Intelligent Signal Processing and Communications Systems (ISPACS)*, Dec. 2011, pp. 1–5.

[10]. N. Kouadria, N. Doghmane, D. Messadeg, and S. Harize, “Low complexity DCT for image compression in wireless visual sensor networks,” *Electron. Lett.*, vol. 49, pp. 1531–1532, Nov. 2013.

[11]. S.-H. Lee, M.-K. Chung, S.-M. Park, and C.-M. Kyung, “Lossless frame memory recompression for video codec preserving random accessibility of coding unit,” *IEEE Trans. Consum. Electron.*, vol. 55, no. 4, pp. 2105–2113, Nov. 2009.

[12]. Y. V. Ivanov and D. Moloney, “Reference frame compression using embedded reconstruction patterns for H.264/AVC decoder,” in *Proc. 3rd Int. Conf. Digit. Telecommun., ICDT*,Jun./Jul.2008, pp. 168–173.

[13]. T.-C. Chen, C.-J. Lian, and L.-G. Chen, “Hardware architecture design of an H.264/AVC video codec,” in *Proc. Asia South Pacific Conf. Des. Autom., ASP-DAC*, Jan. 2006.

[14]. Y. Fan, X. Zeng, and S. Goto, “Optimized 2-D SAD tree architecture of integer motion estimation for H.264/AVC,” *IEICE Trans. Electron.*, vol. E94-C, no. 4, pp. 411–418, Apr. 2011.

[15]. *HM 10.0 reference software*.Available: http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/trunk/

[16]. T. Y. Lee, “A new algorithm and its implementation for frame recompression,” *IEEE Trans. Consum. Electron.*, vol. 47, no. 4, pp. 849–854, Nov. 2001.

[17]. Z.-L. He, C.-Y. Tsui, K.-K. Chan, and M. L. Liou, “Low-power VLSI design for motion estimation using adaptive pixel truncation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 669–678, Aug. 2000.

[18]. F. M. Bayer and R. J. Cintra, “DCT-like transform for image compression requires 14 additions only,” *Electron. Lett.*, vol. 48, pp. 919–921, Jul. 2012.

[19]. U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Edirisuriya, “Improved 8-point approximate DCT for image and video compression requiring only 14 additions,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 61, pp. 1727–1740, Jun. 2014.