An Efficient Hardware Architecture for H.264 Transform and Quantization Algorithms

Logashanmugam.E* , Ramachandran.R**
* Sathyabama University, Chennai, India
** Sri Venkateshwara college of Engineering, Chennai, India

Abstract - In this paper, we present a high performance, low cost and low power hardware architecture for real-time implementation of forward transform and quantization and inverse transform and quantization algorithms used in H.264 / MPEG4 Part 10 video coding standard. The proposed hardware implementation is based on a reconfigurable datapath with only one multiplier. This hardware is designed to be used as part of a complete low power video coding system for portable applications. The proposed hardware architecture is implemented using hardware description language (VHDL). The code is synthesized using Xilinx tool and downloaded into Xilinx FPGA to verify the functionality. The maximum frequency of operation of architecture is about 120 MHz. The FPGA implementation can code 39 VGA frames (640x480) per second.

Index Terms: H.264, Video Compression, Reconfigurable Data path, ASIC, CABAC etc.

1. INTRODUCTION

Video compression systems are being used in different commercial products, from consumer electronic devices such as digital camcorders, cellular phones to video teleconferencing systems. These applications trigger the designers to implement efficient hardware video compression devices. A new International standard for video compression is developed to improve the performance of the existing applications and to enable the applicability of video compression to new real-time applications. This new standard is developed with the collaboration of ITU and ISO standardization organizations, offering significantly better video compression efficiency than previous International Standards and it is called in two different names namely H.264 and MPEG4 Part 10.

The video compression efficiency is achieved in H.264 standard is not a result of any single feature but rather a combination of a number of encoding tools. Two of these tools are the transform and quantization algorithm.
The previous video coding standards, e.g. MPEG-1, H.261, MPEG-2, H.263 and MPEG-4, use the 8x8 Discrete Cosine Transform (DCT) to transform the residual data, where as H.264 uses a 4x4 integer transform for transforming residual data. The integer transform achieves very similar results to 8x8 DCT without any floating point operations. Using low cost binary shifters, all the multiplication operations in the forward and inverse transform algorithms can be implemented in hardware. Since the inverse transform in H.264 is defined by exact integer operations, the mismatches in the inverse transform are avoided. In the quantization algorithm a scaling factor is used which is implemented using a multiplier [3, 4, and 5].

In this paper, we present a high performance, low cost and low power hardware architecture for real-time implementation of H.264 forward transform and quantization and inverse transform and quantization algorithms. The hardware architecture is based on one multiplier with a reconfigurable datapath. We have designed this hardware to be used as part of a complete low power H.264 video coding system for portable applications. The proposed architecture is implemented in VHDL. The VHDL code is verified to work at 120 MHz in a Xilinx Virtex II FPGA [7].

Hardware architecture only for real-time implementation of H.264 forward and inverse transform algorithms is presented in [6]. Our proposed hardware achieves better performance than the previous hardware with less hardware cost. The proposed hardware design is a more cost-effective solution for portable applications. In previous implementation uses 16 adders and 16 internal register files in their datapath as opposed to 3 adders and 6 internal register files in the transform part of our datapath.

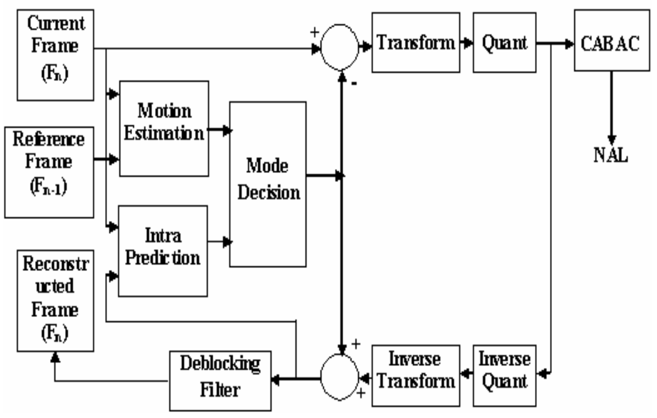


Figure 1 H.264 Encoder Block Diagram

Manuscript received June 5, 2008.
Manuscript revised June 20, 2008.

2. H.264 TRANSFORM AND QUANTIZATION ALGORITHMS OVERVIEW

The basic transform coding process in H.264, shown in Figure 1, is similar to that of previous standards. The process includes a forward transform and quantization followed by zigzag ordering and entropy coding. The transform coded residual data is also reconstructed. The reconstruction process includes an inverse quantization and inverse transform followed by motion compensation. The reconstructed data before deblocking filter is used for intra prediction [14 and 24] in current frame, and the reconstructed data after deblocking filter is used for motion estimation in future entropy coding and reconstruction process in the order shown in figure 1.

The transform and quantization algorithms flow is presented in Figure 2. The input to the forward transform algorithm is a 4x4 block of residual data obtained by subtracting the prediction from the original image data.

The transform and quantization algorithms process the blocks in a macroblock and Send the resulting data to entropy coding and reconstruction process in the order shown in Fig 3.

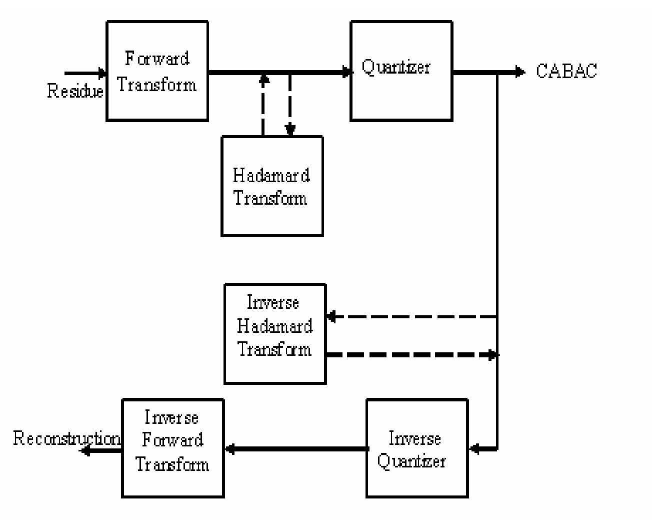


Figure 2 Block Diagram of Transform and Quantization.

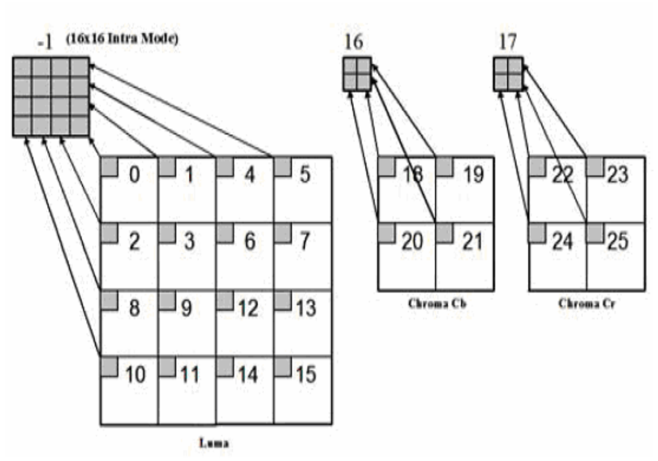


Figure 3 Processing Order of Blocks in a Macroblock

2.1 Overview of Transform Algorithm

H.264 transform algorithm uses four different transform matrices as shown in Figure 4. (4x4 forward integer, 4x4 hadamard, 2x2 hadamard, and 4x4 inverse integer) [3, 4 and 5]. Since 4x4 and 2x2 hadamard transform matrices are symmetric, inverse hadamard transform matrices are same as forward hadamard transform matrices.

$$\begin{bmatrix}1&1&1&1\\2&1&-1&-2\\1&-1&-1&1\\1&-2&2&-1\end{bmatrix}\begin{bmatrix}x0&x1&x2&x3\\x4&x5&x6&x7\\x8&x9&x10&x11\\x12&x13&x14&x15\end{bmatrix}\begin{bmatrix}1&2&1&1\\1&1&-1&-2\\1&-1&-1&2\\1&-2&1&-1\end{bmatrix}$$

(a)

$$\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}\begin{bmatrix}z0&z1&z2&z3\\z4&z5&z6&z7\\z8&z9&z10&z11\\z12&z13&z14&z15\end{bmatrix}\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}$$

(b)

$$\begin{bmatrix}1&1\\1&-1\end{bmatrix}\begin{bmatrix}z0&z1\\z2&z3\end{bmatrix}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$$

(c)

$$\begin{bmatrix}1&1&1&1/2\\1&1/2&-1&-1\\1&-1/2&-1&1\\1&-1&1&-1/2\end{bmatrix}\begin{bmatrix}y0&y1&y2&y3\\y4&y5&y6&y7\\y8&y9&y10&y11\\y12&y13&y14&y15\end{bmatrix}\begin{bmatrix}1&1&1&1\\1&1/2&-1/2&-1\\1&-1&-1&1\\1/2&-1&1&-1/2\end{bmatrix}$$

(d)

Figure 4 Matrices used in H.264 Transform Algorithm

a) 4x4 Forward Integer Transform,
b) 4x4 Hadamard Transform,
c) 2x2 Hadamard Transform,
d) 4x4 Inverse Integer Transform

In the transform coding process, 4x4 integer transform [6] is applied to all the blocks independent of their prediction type and mode. As shown in Figure 3, 4x4 block -1 is formed by the transformed DC coefficients of 4x4 luminance blocks for the macroblocks that are coded in 16x16 Intra mode, and 2x2 blocks 16 and 17 are formed by the transformed DC coefficient for all the macroblocks. After the 4x4 integer transform, 4x4 hadamard transform is applied to block -1 and 2x2 hadamard transform is applied to blocks 16 and 17.
In the reconstruction process, 4x4 inverse hadamard transform is applied to block -1, and 2x2 inverse hadamard transform is applied to blocks 16 and 17. After the inverse hadamard transforms, 4x4 inverse integer transform is applied to all the blocks independent of their prediction type and mode.

2.2. Overview of Quantization Algorithm

A quantization parameter (QP), calculated by the rate control algorithm, is used for determining the quantization step size of transform coefficients in H.264 [3, 4 and 5]. There are 52 quantization parameter values. These values are arranged so that an increase of 1 in quantization parameter means an increase of quantization step size by approximately 12%. An increase of quantization step size by approximately 12% means roughly a reduction of bit rate by approximately 12%.
Quantization of AC coefficients is done by using the equation (1)
$|Zij| = (|Wij|.MF + f) >> qbits$, $sign (Zij) = sign(Wij)$ ------ (1)
Where Wij is the result of forward transformation, MF is a scaling factor, f is a parameter used to avoid rounding errors and it depends on prediction type of the block and QP, *q*bits is a variable depending on QP. Inverse quantization of AC coefficients is done by using the equation (2)
$W'ij = Zij.Vij.2^{floor (QP/6)}$ ------- (2)
Where Zij is the result of forward quantization, Vij are rescaling factors. Quantization of DC coefficients is done similarly.

3. PROPOSED HARDWARE ARCHITECTURE

The proposed hardware architecture includes an input register file, a reconfigurable datapath and its control unit, internal register files and an output register file. The reconfigurable datapath and the register files are shown in Figure 5. The reconfigurable datapath is designed for implementing both forward and inverse transform and quant algorithms. Even though only one multiplier is used in the reconfigurable datapath, the proposed hardware performs forward transform, hadamard transform, quant, inverse hadamard transform, inverse quant and inverse transform at the speed of 120 MHz.

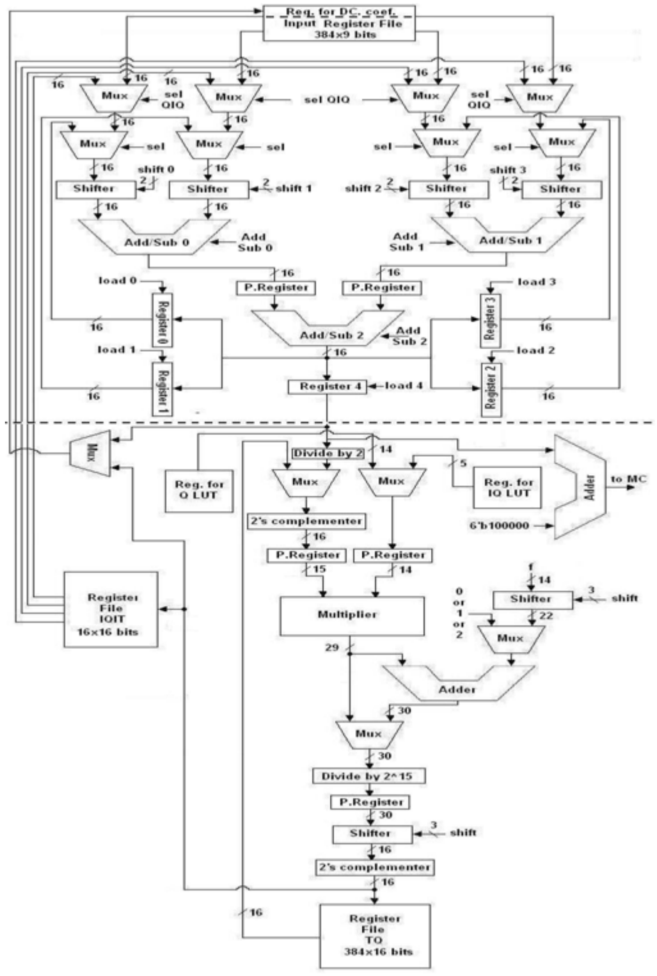


Figure 5 Reconfigurable Data path architecture

In this design we use 384x9 bit input register file to store residual data for a macro block that will be transform coded including both luminance and chrominance blocks. The part of the datapath above the dashed line which is shown in figure 5 performs transform and inverse transform operations. The registers, adders and shifters in this part of the datapath are shared by forward and inverse transform operations. The control unit configures the datapath to perform the forward transform operation when it performs forward transform. When it is used to perform inverse transform, the control unit configures the datapath to perform the inverse transform operations.
The first row of multiplexers is used for selecting the proper inputs for transform operations. They select the data

from input register file for forward transform operations and the data from IQIT register file for inverse transform operations. The second row of multiplexers is used for selecting the proper input data for the first and the second matrix multiplications. They select the data from the first row of multiplexers for the first matrix multiplication operations and the data from register 0, register 1, register 2 and register 3 for the second matrix multiplication operations. Shifters are one bit shifters used for shifting left (multiply by 2) for forward transform operations and for shifting right (divide by 2) for inverse transform operations.

Three adder/subtractors are used in the datapath to achieve high performance with low hardware cost. The first column of the result matrix for the matrix multiplication operations shown in Figure 4 (a) can be calculated using the following four equations (3).

$$[(X_0+X_4+X_8+X_{12})+(X_1+X_5+X_9+X_{13})+(X_2+X_6+X_{10}+X_{14}) + (X_3+X_7+X_{11}+X_{15})]$$

$$[2*(X0+X_4+X_8+X_{12})+(X_1+X_5+X_9+X_{13})-(X_2+X_6+X_{10}+X_{14})-2*(X3+X_7+X_{11}+X_{15})]$$

$$[(X_0+X_4+X_8+X_{12})-(X_1+X_5+X_9+X_{13})-(X_2+X_6+X_{10}+X_{14}) + (X_3+X_7+X_{11}+X_{15})]$$

$$[(X_0+X_4+X_8+X_{12})-2*(X1+X_5+X_9+X_{13})+2*(X2+X_6+X_{10}+X_{14})-(X_3+X_7+X_{11}+X_{15})] \text{ ---- (3)}$$

The four values $(X_0+X_4+X_8+X_{12})$, $(X_1+X_5+X_9+X_{13})$, $(X_2+X_6+X_{10}+X_{14})$ and $(X_3+X_7+X_{11}+X_{15})$ are the results of first matrix multiplication and they are used for calculating the first column of the result matrix containing the transform coefficients. Similarly, the equations for calculating the transform coefficients in each remaining column of the result matrix have four common values that are used to calculate the corresponding transform coefficient. Therefore, 16-bit registers register 0, register 1, register 2 and register 3 are used to store these four common values, i.e. the results of first matrix multiplications. This reduces both the number of cycles and the power consumption of both forward and inverse transform operations. The same method is used to implement the other matrix multiplication operations shown in Figure 4.

Since the order of some of the equations used to perform the matrix multiplications for 4x4 and 2x2 hadamard transforms are not important for functional correctness, we have used the order that gives the lowest power consumption. For example, the first column of the result matrix for the matrix multiplication operations for 4x4 hadamard transform shown in Figure 4 (b) can be calculated using the following four equations in the given order.

$$[(Z_0+Z_4+Z_8+Z_{12})+(Z_1+Z_5+Z_9+Z_{13}) + (Z_2+Z_6+Z_{10}+Z_{14}) + (Z_3+Z_7+Z_{11}+Z_{15})]$$

$$[2*(Z_0+Z_4+Z_8+Z_{12})+(Z_1+Z_5+Z_9+Z_{13})-(Z_2+Z_6+Z_{10}+Z_{14}) - 2*(Z_3+Z_7+Z_{11}+Z_{15})]$$

$$[(Z_0+Z_4+Z_8+Z_{12}) - (Z_1+Z_5+Z_9+Z_{13}) - (Z_2+Z_6+Z_{10}+Z_{14}) + (Z_3+Z_7+Z_{11}+Z_{15})]$$

$$[(Z_0+Z_4+Z_8+Z_{12})-2*(Z1+Z_5+Z_9+Z_{13})+2*(Z2+Z_6+Z_{10}+Z_{14}) - (Z_3+Z_7+Z_{11}+Z_{15})] \text{ ---- (4)}$$

When the equations (4) are calculated in the given order, both the operations (addition or subtraction) performed by adder/subtractor 0 and adder/subtractor 1 and their inputs stay the same in first and second cycles and in third and fourth cycles. Since their inputs and the operations they perform stay the same for two consecutive clock cycles, their outputs stay the same as well. This avoids unnecessary switching activity resulting in lower power consumption for both forward and inverse hadamard transforms. Register 4 contains the input data for the quantization and inverse quantization operations. P. Registers are pipelining registers used to achieve 120 MHz clock frequency in a 2V8000ff1152 Xilinx Virtex II FPGA with speed grade 5. Register 4 stores the results of forward or inverse transform operations.

The part of the datapath below the dashed line performs forward and inverse quantization operations. The registers, adders, shifters and the multiplier in this part of the datapath are shared by forward and inverse quant operations. When the hardware is used to perform forward quantization, the control unit configures the datapath to perform the forward quant operations. When it is used to perform inverse quantization, the control unit configures the datapath to perform the inverse quant operations.

The multiplier used in the datapath is a 15x14 unsigned multiplier. Two multiplexers are used for selecting the proper inputs for the multiplier. One of the multiplexers is used to select either a transformed or inverse transformed value coming from register 4 or a quantized value coming from the output register file TQ. The other multiplexer is used to select either a value from quant lookup table or a value from inverse quant lookup table. The adder at the output of the multiplier and the shifter at one of the inputs of the adder are used to avoid rounding errors that can happen during scaling and rescaling operations. The 3-bit shifter at the output of the multiplier is used to perform scaling and rescaling operations depending on the value of q bits parameter. The result of the shift operation is converted into two's complement form and stored in the output register file TQ.

The transform and quant operations are executed in a pipelined manner. After a transform coefficient is computed, in the next cycle, this coefficient is quantized in the quant part of the datapath and a new transform coefficient is computed in the transform part of the datapath. Since only one multiplier is used in the datapath, quant and inverse quant operations cannot be pipelined. After all the transform coefficients in a block are quantized, inverse quantization starts followed by inverse transform.

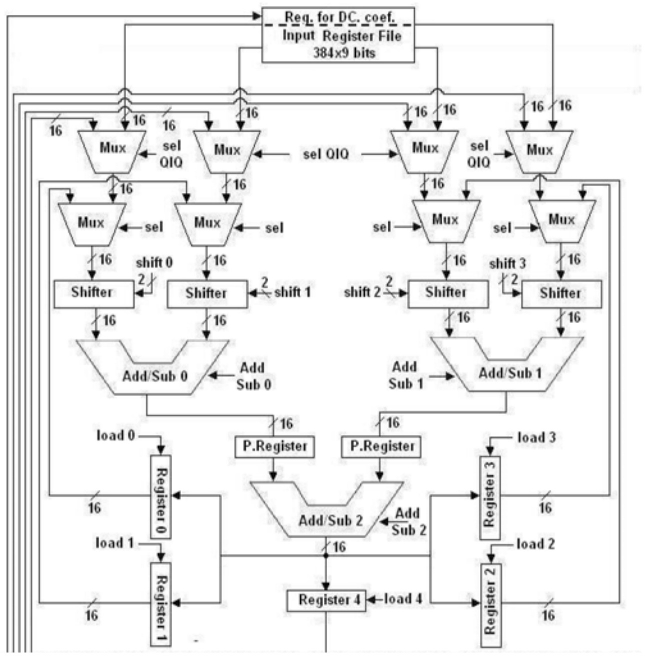


Figure 6 Proposed Transform architecture

The transformation hardware part is modified by replacing adder/subtractor with carry save adder or ripple carry adder, the simulation is carried out for examining the performance of the hardware.

4. SIMULATION AND SYNTHESIS RESULTS

The proposed architecture is coded using VHDL and the code is simulated using Modelsim. The simulation result is shown in the figure 7. We have used built in function of the multiplication which is present in the Xilinx Virtex II FPGA (2V8000ff1152). The simulation is carried out when replacement is done with both carry save adder and ripple carry adder also. The table 2 shows that the comparison analysis of transformation with two different adders. The code is synthesized using Xilinx project navigator and the report is shown in figure 8. The device utilization of the overall architecture is shown in table 1.

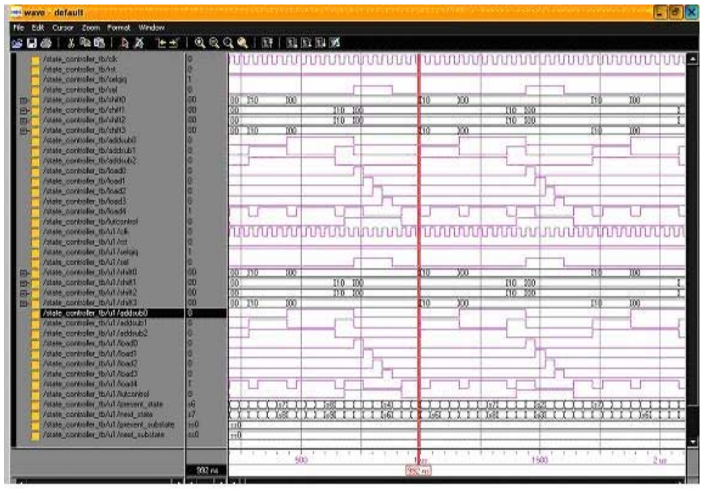


Figure 7 Simulation result of the proposed hardware unit

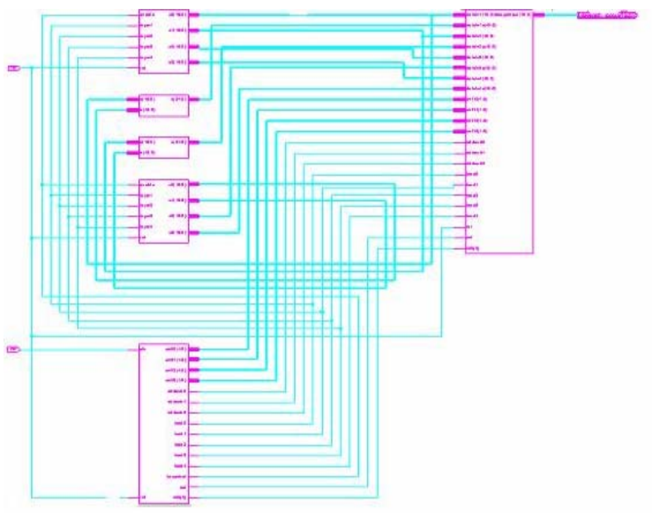


Figure 8 Synthesis report

Table I
Device utilization of Virtex II FPGA (2V8000ff1152)

Sl.No	Description	Used	Available	% used
1	Number of Slices	291	46592	0
2	Number of Slice Flip Flops	228	93184	0
3	Number of 4 input LUTs	546	93184	0
4	Number of bonded IOBs	18	824	2
5	Number of MULTI8X18s	1	168	0
6	Number of GCLKs	4	16	25

Table II
Comparative Analysis of transformation algorithm with two different adders

Sl.No	Parameter	Ripple carry adder	Carry save adder
1	Minimum period	16.061ns	12.537ns
2	Minimum input arrival time before clock	16.866ns	14.725ns
3	Maximum output required time after clock	4.932ns	: 4.932ns
4	Maximum Frequency	62.263MHz	79.764MHz

5. CONCLUSIONS

In this paper, we presented and implemented a high performance and low cost hardware architecture for H.264 forward transform and quantization and inverse transform and quantization algorithms. The hardware is designed such that the datapath is reconfigurable and it uses only one multiplier.

This hardware design is aimed for portable applications since it is a complete low power H.264 video coding system. The transformation architecture with carry save adder gives better performance. The overall architecture is improved with the previous architecture which is running at 81 MHz. The speed of operation increases at 48% compared with the previous implementation. The modified architecture operates at 120 MHz. The FPGA implementation can code 39 VGA frames (640x480) per second.

The future work of this research is to implement in ASIC to further reduce area and increase the speed of operation.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. on Circuits and Systems for Video Technology vol. 13, no. 7, pp. 560–576, July 2003

[2] I. Richardson, H.264 and MPEG-4 Video Compression, Wiley, 2003

[3] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003

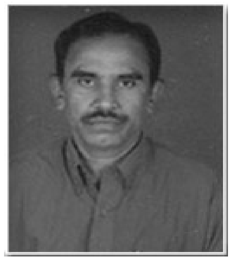
[4] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Joint Model (JM) Reference Software Version 9.2, http://bs.hhi.de/suehring/

[5] H. Malvar, A. Hallapuro, M. Karczewicz. and L. Kerofsky, "Low-Complexity Transform and Quantization in H.264 / AVC", IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 598–603, July 2003.

[6] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen, "Parallel 4x4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC / H.264", Proc. of IEEE ISCAS, 20

[7] Xilinx Inc., Virtex-II™ Platform FPGAs: Complete Data Sheet DS031, http://www.xilinx.com, March 2004
[8] D. Marpe, G. Blattermann, G. Heising, and T.Wiegand, "Further Results for CABAC Entropy Coding Scheme", Austin, TX, ITU-T SG16/Q.6 Doc. VCEG-M59, 2001.
[9] D. Marpe, G. Blattermann, and T. Wiegand, "Improved CABAC", Pattaya,Thailand, ITU-T G16/Q.6 Doc. VCEG-O18, 2001.
[10] D. Marpe, G. Blattermann, T. Wiegand, R. Kurceren, M. Karczewicz and J. Lainema, "New results on improved CABAC", in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVTB101,Geneva, Switzerland, Feb. 2002.
[11] H. Schwarz, D. Marpe, G. Blattermann, and T. Wiegand, "Improved CABAC," in Joint Video Team of ISO/IEC JTC1/SC29/WG11& ITU-T G16/Q.6 Doc. JVT-C060, Fairfax, VA, Mar. 2002.
[12] D. Marpe, G. Heising, G. Blattermann, and T. Wiegand, "Fast arithmetic coding for CABAC," in Joint Video Team of ISO/IECJTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-C061, Fairfax, VA, Mar. 2002.
[13] H. Schwarz, D. Marpe, and T. Wiegand, "CABAC and slices", in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc.JVT-D020, Klagenfurt, Austria, July 2002.
[14] M. Karczewicz, "Analysis and simplification of intra prediction", in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-D025, Klagenfurt, Austria, July 2002.
[15] D. Marpe, G. Blattermann, G. Heising, and T. Wiegand, "Proposed cleanup changes for CABAC," in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-E059, Geneva, Switzerland, Oct. 2002.
[16] F. Bossen, "CABAC cleanup and complexity reduction," in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVTE086, Geneva, Switzerland, Oct. 2002.
[17] D. Marpe, H. Schwarz, G. Blattermann, and T.Wiegand, "Final CABAC clean up," in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-F039, Awaji, Japan, Dec. 2002.
[18] D. Marpe and H. L. Cycon, "Very low bit-rate video coding using wavelet-based techniques," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 85–94, Apr. 1999.
[19] G. Heising, D. Marpe, H. L. Cycon, and A. P. Petukhov, "Wavelet-Based very low bit-rate video coding using image warping and overlapped block motion compensation," Proc. Inst. Elect. Eng.—Vision, Image and Signal Proc., vol. 148, no. 2, pp. 93–101, Apr. 2001.
[20] S.J. Choi and J.W.Woods, "Motion compensated 3-D subband coding of video," IEEE Trans. Image Processing, vol. 8, pp. 155–167, Feb. 1999.
[21] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 243–250, June 1996.
[22] D. Marpe and H. L. Cycon, "Efficient pre-coding techniques for wavelet based image compression," in Proc. Picture Coding Symp., 1997, pp. 45–50.
[23] J. Rissanen and G. G. Langdon Jr, "Universal modeling and coding," IEEE Trans. Inform. Theory, vol. IT-27, pp. 12–23, Jan. 1981.
[24] J. Rissanen, "Universal coding, information, prediction, and estimation," IEEE Trans. Inform.Theory, vol. 30, pp. 629–636, July 1984.
[25] W. B. Pennebaker and J. L. Mitchell, JPEG: Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1993.
[26] A. Papoulis, Probability, Random Variables, and Stochastic Processes, New York: McGraw-Hill 1984, pp. 37–38.

E.Logashanmugam received Masters in Electronics engineering from Anna University. Currently he is doing PhD in Sathyabama University. He is working as Head of Department of Electronics and Communication in Sathyabama University. His field of interest is Image processing and VLSI Design .He is a member IEEE and ISTE Email: logu999@yahoo.com



Dr.R.Ramachandran received PhD from Anna University. He is working as Principal in Sri Venkateswara College of Engineering. His field of interest is Image processing and VLSI Design .He is a member IEEE and Life fellow of IETE.