An Implementation of High Speed DCT and Hadamard Transform for H.264

Bui An Dong and Huynh Quoc Thinh

Faculty of Electronics and Telecommunications, HCMC University of Science, Vietnam
badong@hcmus.edu.vn

Abstract. The focus of this paper is the improvement of performance of the high speed forward integer DCT architecture for H.264 and Hadamard for luma in intra prediction. The DCT and Hadamard architecture are the same architectures, a serial architecture and pipeline processing, includes two 1-D architectures. Each architecture uses six adders and four shifters (DCT) or one shifter (Hadamard). They can operate simultaneously for high-throughput processing. The proposed DCT and Hadamard algorithms are verified by Matlab and VCS tool of Synsopsys. Then the design is synthesized by Design Compiler with 90nm CMOS technology. The DCT and Hadamard core requires only 5103 logic cells (DCT core uses 2170 logic cells; Hadamard core uses 2392 logic cells, and other cores use 540 logic cells) and needs 24 clock cycles to finish one 4x4 DCT or Hadamard block at pipeline mode. Its frequency can operate at 250MHz.

Keywords: DCT, Hadamard, H.264, HDTV, MPEG.

1 Introduction

H.264, an advanced compression standard currently being developed by the JVT (Joint Video Team) and was standardized in 2003, is widely used as: HDTV, mobile digital TV, satellite TV, TV conference ... By the transmission bit rate can be reduced by 50% compared with the previous standards, H.264 requires high complexity of encoding and decoding system. In addition to other complex algorithms, H.264 uses the Integer Discrete Cosine transform. It is performed on integer more accurate and faster than the calculation of the real numbers with floating point or static point.

The parallel DCT architecture is high-speed. In contrast, the serial DCT architecture saves more power. Today, mobile applications require high speed and low-power. Therefore, this paper proposes a new serial DCT architecture. These architectural design test algorithms in Matlab and the results are compared with hardware design simulate in Synopsys tools with process 90nm of Synopsys.

As a result, the DCT hardware resources are reduced more than 58% compared with parallel DCT architecture [5] and its clock cycle is reduced more than 14.6% compared with serial DCT architecture [3]. Besides, Hadamard hardware resources are less then approximate 60% compared with architecture [6] and speed increases of 2.5 times and 1.7 times compared with architecture [6] and [8].

I. Zelinka et al. (eds.), *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*, Lecture Notes in Electrical Engineering 282,
DOI: 10.1007/978-3-642-41968-3_34,

This paper is organized as follows: in section 2, the background of DCT transforms and Hadamard transforms for 16x16 block in intra prediction, as supported by the H.264 standard and the hardware architecture of forward DCT and Hadamard transforms. Results of simulation and synthesis are described in section 3. Section 4 is conclusions.

2 Experimental

2.1 Algorithms

The 4x4 forward DCT transforms is defined as:

$$Y_{ij} = C\ X\ C^T \tag{1}$$

Where X is a 4x4 pixel block; C is:

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$$

The 4x4 forward DCT transform is used for all of intra and inter mode. Similarly, the 4x4 forward Hadamard transform is only used for luma mode in intra prediction and it is defined as:

$$K_{ij} = (K\ M\ K^T)/2 \tag{2}$$

Where M is 4x4 DC block, includes DC values which is transformed DCT in luma mode of intra prediction; K is:

$$K = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

2.2 Experimental

To further simplify the computation, the scalar multiplication is integrated into the quantization unit in H.264/AVC. We have only to implement CXC^T in the transform engine.

The 2D-DCT is separated into two 1D-DCTs. Firstly, the input data is transformed by column, and then the obtained results are transposed. Finally, they are transformed again by row.

Consider 1D-DCT and 2D-DCT as figure 1. The transform is separate from two stages: buffer stage and record stage. The algorithm is shown below figure with one node is an addition, and the multiplication with 2 is replaced by a shifter.

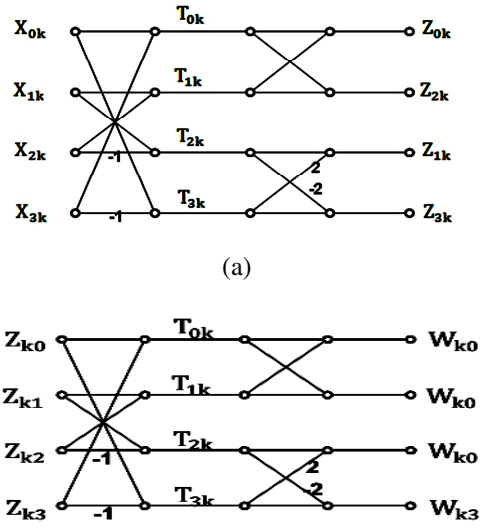


Fig. 1. (a) The algorithm of 1-D DCT (b) The algorithm of 2-D DCT

Figure 2 shows the 1D-DCT architecture, includes the ping-pong buffer, the adder-subtractors, the sign-extended and the left-shifter.

The ping-pong buffer holds data while input data is being written serially. The adder-subtractors are built from carry look ahead adder to get high speed. The shifters operate as a multiplier by 2. The sign-extended convert signed 9-bit data to sign 12-bit data.

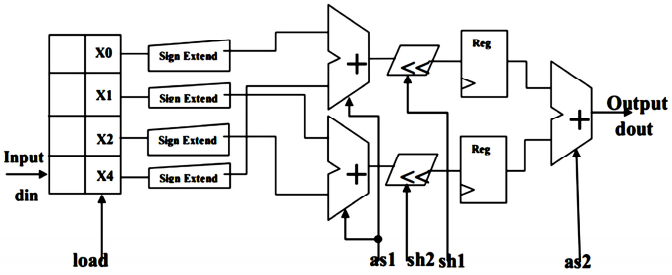


Fig. 2. The 1D-DCT architecture

We can use 1D-DCT to compute second dimension of DCT by a multiplexor. This saves area and power, but speed will be decreased a half. To improve throughput, we can construct a pipeline architecture as figure 3. The 1D-DCT and 2D-DCT can work simultaneously so speed of the circuit will be increased two times.

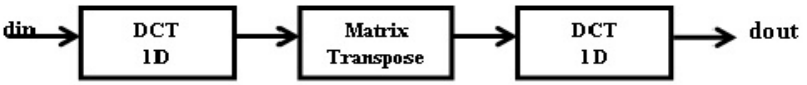


Fig. 3. The system of 2D-DCT

Ping-pong Buffer. The ping-pong buffer (figure 4) is the array of registers with serial input data to be converted to parallel data. When the rising edge of the clock, serial input data is written to the registers Reg_1, Reg_2 and Reg_3. When the load signal to 1, it enables the registers Reg_4, Reg_5, Reg_6 and parallel Reg_7 record and hold data until new data are written to. A special feature of this register array is able to calculate the value maintained while data continues to be included the serial.

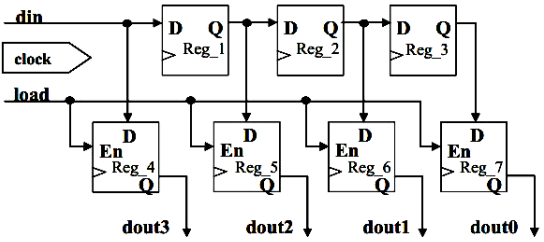


Fig. 4. Ping-pong Buffer

Adder-Subtractor. The adder-Subtractor, Carry Look-Ahead Adder (CLA), is not only high speed but also low power. We group 3 or 4 bits into a small block; carry number of each block is calculated directly from the Pi, Gi of block and it is Cin for next block. At the 3-bit CLA block, in Figure 5 (a), C1, C2 are calculated by CLA, and Cout is calculated by P1, P2, G1, G2 and Cin without waiting for the C1 and C2. While the first 3-bit block is calculating, C3 is calculated from Cin, C6 is calculated from C3 and etc.... Therefore, the Adder performs faster.

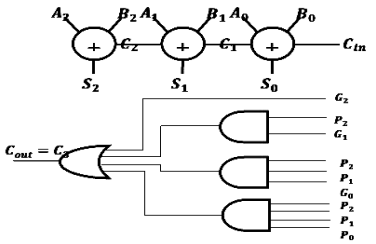


Fig. 5. (a)3-bit CLA Adder

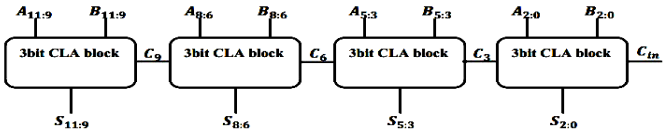


Fig. 5. (b)12-bit CLA Adder

Left Shifter. The left shifter is same multiply 2. When sh = 0, data passes through, and when sh = 1, the shifter performs 1-bit left shift.

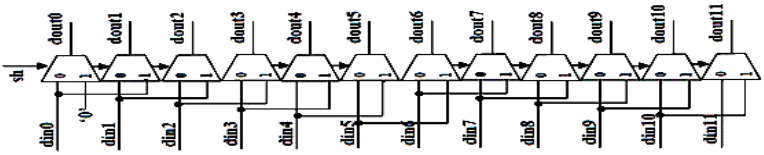


Fig. 6. Shifter

Matrix Transpose. The Matrix Transpose in Figure 7 is two parallel register arrays, each array is designed according to the serial input or output data. When the load signal is 1, t registers is received values corresponding to the r registers. Finally, the data in t registers is shifted and put out sequentially. The matrix transpose needs 16 cycles to receive serial data, 1 cycle to transpose and 16 cycles to put out the serial data.

The advantage of this matrix transpose is able to write and put out data in the same time. This improves speed but it needs more resource than the RAM transpose because the memory cell of the RAM is less transistor than the register, uses Flip-flop.

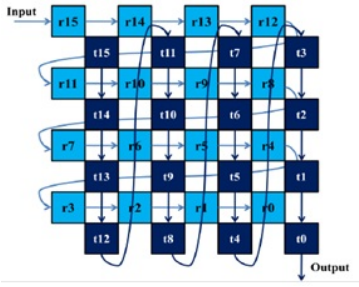


Fig. 7. Matrix Transpose

Operation. The datapath of 1D-DCT is divided into two-stage correlations with the algorithm. In the first four clock cycles, data is entered into the first register array of ping-pong buffer. The next cycle, data is simultaneously loaded into remain the next register array of ping-pong buffer and held during these cycles. In these cycles, the circuit operates as follows

- The 6^{th} cycle, the 1-D DCT calculates and records result into register,
- The 7^{th} cycle, the 2-D DCT calculates and outputs the first result. In the also 7^{th} cycle, the 1-D DCT calculates the second data and records into register.
- The process is continued until completing a 4x4 block. This circuit takes 21 cycles to finish.

The process is indicated in the table 1.

Table 1. The operation of DCT architecture

Cycles	Operation	Cycles	Operation
0	Reset	11	T_{01}, T_{11},W_{11}
1	Input x_{00}	12	T_{21}, T_{31}, W_{21}
2	Input x_{10}	13	T_{02}, T_{12}, W_{31}
3	Input x_{20}	14	T_{22}, T_{32}, W_{02}
4	Input $x_{30,}$ Load data to the second floor of ping-pong register	15	T_{02}, T_{12}, W_{12}
5	T_{00}, T_{10}	16	T_{22}, T_{32}, W_{22}
6	T_{20}, T_{30}, W_{00}	17	T_{02}, T_{12}, W_{32}
7	T_{00}, T_{10}, W_{10}	18	T_{23}, T_{33},W_{03}
8	T_{20}, T_{30}, W_{20}	19	T_{03}, T_{13}, W_{13}
9	T_{01}, T_{11}, W_{30}	20	T_{23}, T_{33},W_{23}
10	T_{21}, T_{31},W_{01}	21	T_{03}, T_{13}, W_{33}

Because the Hadamard transform for luma in intra prediction is calculated similarly DCT transform, the Hadamard architecture and operating are similarly DCT architecture and operating. The 16 DC values, after DCT calculation, are written to another DC matrix to calculate Hadamard, in figure 8.

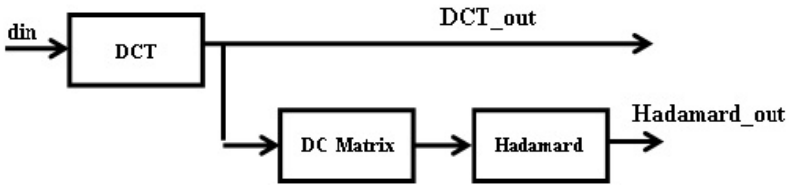


Fig. 8. DCT and Hadamard Architecture

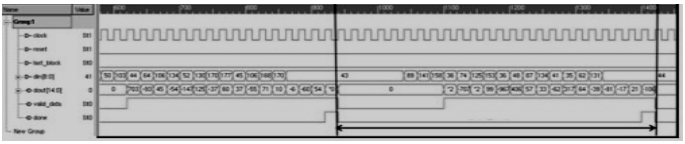
These values put in Hadamard transforms to calculate and the Hadamard transform needs also 21 cycles to finish. At this time, DCT transforms is still processing.

3 Results

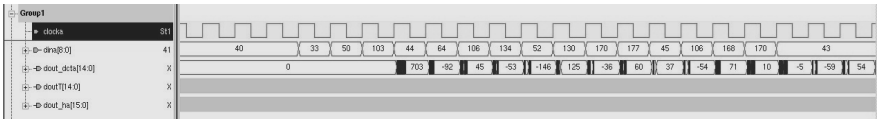
Data after prediction, estimation and motion compensation creates a residual matrix with one element is a signed 9-bit. This is also the input of DCT.

$$\begin{bmatrix} 41 & 32 & 32 & 50 \\ 37 & 43 & 44 & 29 \\ 56 & 48 & 38 & 58 \\ 49 & 43 & 49 & 74 \end{bmatrix} \xrightarrow{\text{DCT}} \begin{bmatrix} 703 & -93 & 45 & -54 \\ -147 & 125 & -37 & 60 \\ 37 & -55 & 71 & 10 \\ -6 & -69 & 54 & -100 \end{bmatrix}$$

The DCT result simulated by Matlab is matched with VCS and Design Compiler in figure 9 and figure10:



(a)



(b)

Fig. 9. The result of 2D-DCT is simulated by VCS: (a) Before synthesis (b) After synthesis

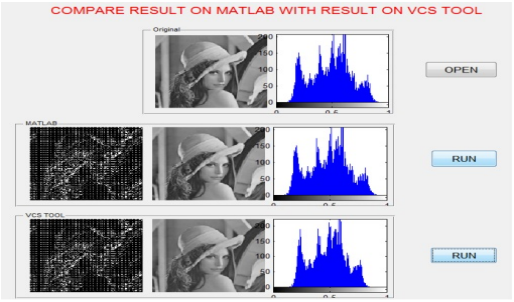
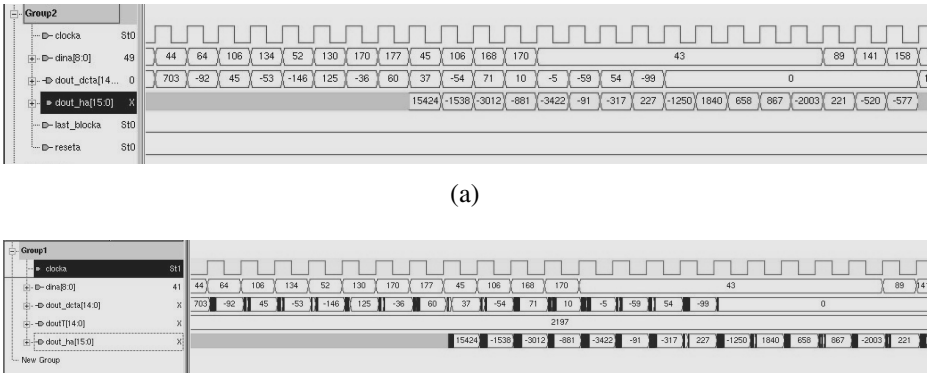


Fig. 10. Compare the results on Matlab and results on Modelsim

The Hadamard result, simulated by Matlab, is matched with VCS and Design Compiler in figure 11:

$$\begin{bmatrix} 703 & 1592 & 1393 & 686 \\ 894 & 1997 & 2684 & 2053 \\ 1075 & 2682 & 2810 & 2480 \\ 2324 & 2619 & 2660 & 2197 \end{bmatrix} \xrightarrow{\text{Hadamard}} \begin{bmatrix} 15425 & -1538 & -3013 & -882 \\ -3423 & -92 & -318 & 227 \\ -1250 & 1840 & 659 & 867 \\ -2004 & 222 & -520 & -578 \end{bmatrix}$$



(a)

(b)

Fig. 11. The result of Hadamard is simulated by VCS: (a) Before synthesis (b) After synthesis

The circuit is synthesized base on Design Compiler and generic 90nm library of Synopsys at 250MHz frequency, follow a top-down method. Condition of the circuit: Vdd=0.7V, T=125oC. The result of the top module:

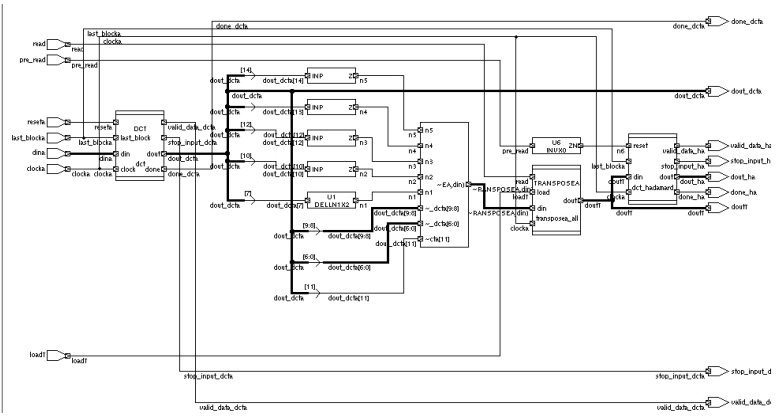


Fig. 12. The DCT and Hadamard datapath for luma mode in intra prediction

Table 2. The characteristic of design

Characteristic	frequency	Leaf Cell	Cell area	Circuit area	Dynamic power	Leakage power
Unit	MHz		μm^2	μm^2	µW	µW
Value	250	5103	83278	89127	130.1335	327.157

This work constructs a serial architecture combine with pipeline to increase speed of calculation. The DCT circuit needs 21 clock cycles to finish 4x4 residual block. The DCT system includes in 6 adder-subtractor 4 shifter so the area and power will be improved.

Table 3. Comparison the DCT design with other designs

Reseach	Architecture	No. of clock	No. of gate or cell	Component of datapath	
[3]	Serial	164	1189	1 add-sub, 2 Mux, 1 shifter, rom 16x16	Non-pipeline
[4]	parallel	12	FPGA	10 add-sub, 4 shifter	pipeline
[5]	parallel	8	3737		
DCT in this work	serial	24	2170 (leaf cell)	6 add-sub, 4 shifter	pipeline

4 Conclusion

This paper has analyzed and constructed a serial and pipeline architecture of DCT and Hadamard transform for luma in intra prediction. The system is described by Verilog HDL, simulated, synthesized base on 90nm library. The architecture requires 5103 logic cells and 21 clock cycles to calculation DCT or Hardamard for a 4x4 block at pipeline mode. Its frequency can operate at 250MHz and power is 327μW. Our DCT hardware resources are reduced more than 58% compared with parallel DCT architecture [5] and its clock cycle is reduced more than 14.6% compared with serial DCT architecture [3]. Our Hadamard hardware resources are less then approximate 60% compared with architecture [6] and speed increases of 2.5 times and 1.7 times compared with architecture [6] and [8].

References

1. Lin, Y.-L.S., Kao, C.-Y., Kuo, H.-C., Chen, J.-W.: VLSI Design for Video Coding - H.264/AVC Encoding from Standard Specification to Chip. Springer, Heidelberg (2010) ISBN 978-1-4419-0958-9
2. Hallapuro, A., Karczewicz, M.: Low Complexity Transform and Quantization – Part I: Basic Implementation, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) (2002)
3. Prasoon, A.K., Rajan, K.: 4×4 2-D DCT for H.264/AVC. In: International Conference on Advances in Computing, Communication and Control, ICAC3 2009 (2009)
4. Sun, S., Qi, H.: A Pipelining Hardware Implementation of H.264 Based on FPGA. In: International Conference on Intelligent Computation Technology and Automation (2010)
5. Wang, T.-C., Huang, Y.-W., Fang, H.-C., Chen, L.-G.: Parallel 4X4 2D trans-form and inverse transform architecture for MPEG-4 AVC/H.264. In: Proceedings of the 2003 International Symposium on Circuits and Systems, vol. 2, pp. II-800–II-803 (2003)
6. Chen, L.-F., Li, K.-H., Huang, C.-Y., Lai, Y.-K.: Analysis and architecture design of multi-transform architecture for h.264/avc intra frame coder. In: ICME (2008)
7. Tran, X.-T., Tran, V.-H.: An Efficient Architecture of Forward Transforms and Quantization for H.264/AVC Codecs. REV Journal on Electronics and Communications 1(2) (April–June 2011)

8. Huang, Y.Q., Liu, Q., Ikenaga, T.S.: Highly parallel fractional motion estimation engine for super hi-vision 4kx4k@60lps. IEEE Int. Works. Multimedia Signal Processing (2009)
9. Malvar, H.S., Hallapuro, A., Karczewicz, M., Kerofsky, L.: Low-Complexity Transform and quantization in H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology 13(7) (2003)
10. Agostini, L., Porto, R., Porto, M., Silva, T., Rosa, L., Guntzel, J., Silva, I., Bampi, S.: Forward and Inverse 2-D DCT Architechtures Targeting HDTV For H.26/AVC Video Compression Standard. Latin American Applied Research (2007)
11. EL-Banna, H., EL-Fattah, A.A., Fakhr, W.: An Efficient Implementation of the 1D DCT using FPGA Technology. In: Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2004), pp. 356–360 (2004)
12. Xiuhua, J., Caiming, Z., Yanling, W.: Fast Algorithm of the 2-D 4x4 Inverse Integer Transform for H.264/AVC. In: IEEE Innovative Computing, Information and Control International Conference, ICICIC (2007)
13. Owaida, M., Koziri, M., Katsavounidis, I., Stamoulis, G.: A High Performance and Low Power Hardware Architecture for The Transform & Quantization Stages In H.264. In: ICME (2009)
14. Liu, Z., Wang, D., Ikenaga, T.: Hardware optimizations of variable block size hadamard transform for h.264/avc frext. In: ICIP (2009)