Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 7, July 2015, pg.67 – 73

RESEARCH ARTICLE

IMPLEMENTATION OF LOEFFLER ALGORITHM FOR LENGTH 2^N IDCT

Satya Parakala
M.Tech (VLSI System Design)
Jits, karimnagar
parkala.vasu@gmail.com

M. Sagar
Associate Professor
Jits, karimnagar
sagarnettu34@gmail.com

Abstract—In this paper we investigate hardware implementations of 8x8 DCT and IDCT on different FPGA technologies using the modified Loeffler algorithm. The investigations involved simulations and synthesis of VHDL code utilizing recent FPGA families of Xilinx, Altera, and Lucent. We aimed at achieving the most demanding real-time requirements of some standardized frame resolutions and rates. Synthesis results for 8-point DCT/IDCT implementations indicate operating frequencies of 50 MHz, 60 MHz, and 22 MHz for the investigated Xilinx, Altera and Lucent FPGA chips, respectively. These frequencies allow 2193 SIF and 100 HDTV frames to be processed by the Xillinx FPGA. The resulting frame processing rates for Lucent are 877 and 40 for SIF and HDTV, while for Altera they are 647 and 29, respectively. Results indicate that the investigated FPGA implementations would speed DCT based compression algorithms up to frame rates well above the real-time requirements of SIF, CCIR-TV and HDTV frame formats.

Keywords— Discrete Cosine Transform (DCT); VHDL; FPGA; Reconfigurable Processor

I. INTRODUCTION

The discrete cosine transform (DCT) and the inverse discrete cosine transform (IDCT) are substantial performance bottlenecks in the contemporary visual data compression algorithms (JPEG, MPEG, etc.). Implementing DCT/IDCT, as an ASIC is a design solution, which meets the real time processing requirements, but it lacks flexibility. Another, more flexible solution, still capable to achieve real-time performance, is the reconfigurable realization of the transforms. Such DCT/IDCT implementations mapped on FPGAs will be discussed here. In this paper we investigate implementations of 8x8 DCT and IDCT hardware units mapped on various FPGA technologies using the modified Loeffler algorithm [1], [2]. We aimed at achieving the most demanding real-time requirements of some standardized frame resolutions such as the Source Input Format (SIF) and the International Consulting Committee on Radio and Television (CCIR-TV). Our particular interest was in performance improvements for the High Definition Television (HDTV) standard. The investigation involved generation, simulation and synthesis of VHDL code using *ModelSim*TM and *Leonardo Spectrum*TM as design environments. During the design process we

used VHDL libraries for the recent FPGA families of Xilinx, Altera and Lucent. Synthesis results for an 8-bit IDCT implementation indicate:

• 214 Configurable Logic Block slices and 22 multipliers in Xilinx Virtex II Technology; 1482 Altera Acex-1K Logical Cells; 1488 Lucent's Orca Look-UP Tables.

• Operating frequencies of 50 MHz for Xilinx, 60 MHz for Altera, and 22 MHz for Lucent.

• 2193 SIF and 100 HDTV frames per second to be processed by Xillinx Virtex II FPGA; 877 SIF and 40 HDTV frames per second processing speed by Lucent's Orca; 647 SIF and 29 HDTV frames per second throughput by Altera's Acex. Synthesis and simulation results prove that the investigated FPGA implementations can speed up DCT to frame rates well above the real-time requirements of SIF, CCIR-TV and HDTV.

DCT and IDCT have been widely used in video data compression standards. The decorrelation and energy compaction properties of the transform have been exploited to achieve high compression ratios in MPEG and JPEG

$$X(k) = \frac{2}{N} c_k \sum_{n=0}^{N-1} y(n) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \; k = 0,1,\ldots, N-1;$$

and the N-point 1-D IDCT is defined by:

$$y(n) = \frac{2}{N} \sum_{k=0}^{N-1} c_k X(k) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \; n = 0,1,\ldots, N-1;$$

where $c_k = \begin{cases} 1/\sqrt{2}, k = 0 \\ 1, k \neq 0 \end{cases}$.

DCT and IDCT are highly computational intensive, which creates prerequisites for performance bottlenecks in systems utilizing them. To overcome this problem, a number of algorithms have been proposed for more efficient computations of these transforms. In our experiments we use an 8-point 1-D DCT/IDCT algorithm, proposed by van Eijdhoven and Sijstermans [1]. This algorithm is a slight modification of the original Loeffler algorithm [2], which provides one of the most computationally efficient 1-D DCT/IDCT calculations. The modified Loeffler algorithm for calculating 8-point 1-D IDCT is illustrated in Figure 1.

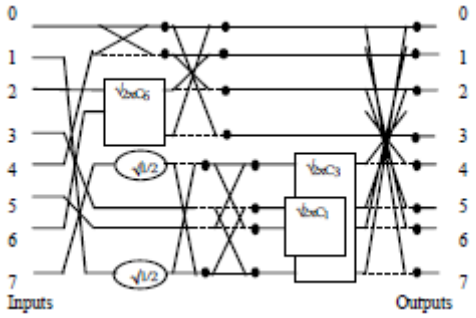
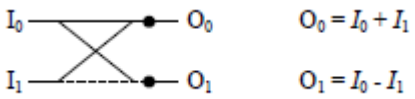


Figure 1 The 8-point IDCT - modified Loeffler algorithm.

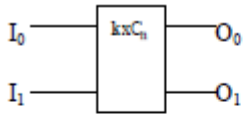
The round block in Figure 1 signifies a multiplication by √1/2. The butterfly block and the equations associated to it are presented in Figure 2



,,

Figure 2 The Butterfly

The rectangular block depicts a rotation, which transforms a pair of inputs [I0, I1] into outputs [O0, O1]. The symbol and associated equations are depicted in Figure 3.



$$O_0 = I_0 k\cos\left[\frac{n\pi}{16}\right] - I_1 k\sin\left[\frac{n\pi}{16}\right] = C'_n I_0 - S'_n I_1$$

$$O_1 = I_0 k\sin\left[\frac{n\pi}{16}\right] + I_1 k\cos\left[\frac{n\pi}{16}\right] = S'_n I_0 + C'_n I_1$$

Figure 3 The rotator and its associated equations

The implementation of the rotator depicted in Figure 4 utilizes four multipliers and two adders to shorten critical path and improve numerical accuracy. This direct implementation has been proven to be ideal for fixed point arithmetic [5]. Indeed, some other implementations of the rotator are possible, e.g., with three multipliers and three adders. These alternative designs, however, have longer critical paths and involve initial additions, which may lead to overflows and may affect the accuracy of the calculations.

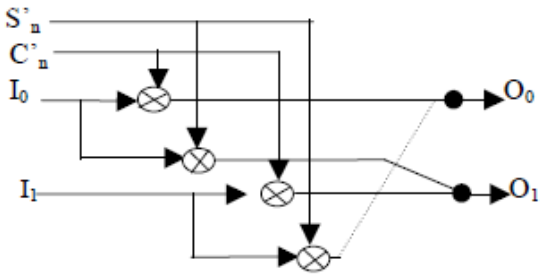


Figure 4 Implementation of the rotator for IDCT

We depict the algorithm of 8-point DCT in Figure5

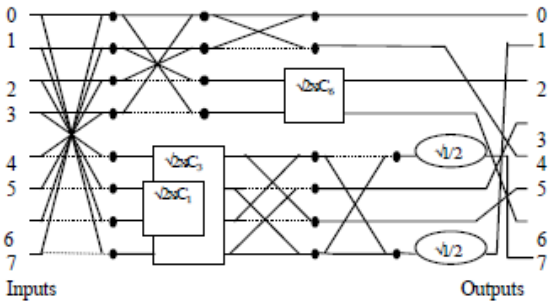


Figure 5 The 8-point DCT - modified Loeffler algorithm

The functionality of the rotator in DCT is slightly different than in IDCT, while the round block and the butterfly are exactly the same. The DCT rotator block equations are:

$$O_0 = I_0 k\cos\left[\frac{n\pi}{16}\right] - I_1 k\sin\left[\frac{n\pi}{16}\right] = C'_n I_0 - S'_n I_1$$

$$O_1 = -I_0 k\sin\left[\frac{n\pi}{16}\right] + I_1 k\cos\left[\frac{n\pi}{16}\right] = -S'_n I_0 + C'_n I_1$$

In video data compression standards, the 2-D DCT/IDCT is defined. One possible approach to compute the 2-D DCT/IDCT is the standard row-column separation. In this approach, the 1-D transform is applied to each row. On

each column of the result 1-D transform is performed again, to produce the final result of the 2-D DCT/IDCT. In our experiments we use this strategy.

II. METHODOLOGY OF THE EXPERIMENTATION

Our experiments involve processing video data with different frame formats. We have chosen the SIF, CCIRTV and the HDTV formats, since they have been considered by many video compression standards. The frame resolutions for SIF, CCIR-TV and HDTV are 352x288, 525x720 and 1152x1926, respectively. We have written synthesizable VHDL models of two units, one describing 1-D DCT and the other – 1-D IDCT. The designs have been implemented according to the modified Loeffler algorithm. We both simulated and synthesized the VHDL models for three different FPGA technologies, namely Virtex II, Acex-1K and Orca using the following design tools:

• *ModelSim*TM SE/EE from Model Technology, version 54.b, revision 2000.06, for simulating the VHDL source code;
• *LeonardoSpectrum*TM from Exemplar, version v2000.1a2.75, for the synthesis of VHDL source code. For the design of DCT we considered 8-bit input data for consistency with the 8-bit color presentation in visual data compression standards like MPEG and JPEG. The output data width was designed to be 10-bit. Similarly, 10-bit inputs and 8-bit outputs were considered for the IDCT design. The row-column separation strategy was used to compute the 2-D DCT/IDCT. As we have used 8-point 1-D DCT and IDCT, the FPGA I/O ports delay, reported by the synthesis software, is in essence the data processing delay for 8 pixels. Implementing matrix transposition without extra delay, we can multiply the 8- point 1-D DCT I/O latency by 16 to calculate the latency of the 8x8 DCT transform. This is in essence the processing latency for one 8x8 pixel block. Given this latency, we can easily calculate the tame, required to transform all 8x8 blocks in any video frame for the
selected formats - SIF, CCIR-TV and HDTV. Frame processing rate (frames per second) of the implemented DCT/IDCT was the main criterion used to estimate and compare the FPGA mappings on the three different technologies.

III. EXPERIMENTAL RESULTS

Synthesis results for 8-point DCT and IDCT units are included in Table I. These results indicate that the Xilinx FPGA implementations of DCT/IDCT can process higher numbers of frames per time unit, compared to the other two FPGA technologies. One reason for this considerable data processing speed is the utilization of coarse-grain reconfigurable resources available in the Virtex II FPGA. In particular, the usage of hardwired multipliers and fast carry chains lead to a severe acceleration of the implemented computations We were particularly interested whether the FPGA implementations of the designs would be fast enough to meet the real time requirements of the selected video formats. For SIF and CCIR-TV, the requirements for real time processing rates are 25 frames per second. That Xilinx FPGA implementations process the highest number of SIF frames per second. The other two FPGA technologies, using finer-grain resources, although slower, are capable of processing SIF frames at speeds, much higher than the required real-time rates (25 frames per second).

The blocks of algorithm are created using the Xilinx System Generator are finally converted to bit file using the Xilinx ISE. This bit file is programmed on the FPGA kit using the Xilinx Chip Scope programming and debugging tool. The FPGA used is Virtex-II(XC4VSX3510FF668). Below Figure shows the block diagram of the laboratory test bench.

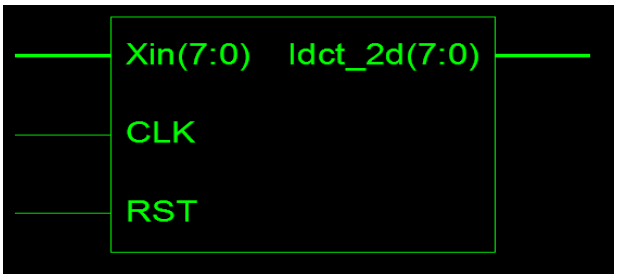


Fig.4.1 Schematic diagram for DCT-IDCT

Inputs of Schematic are Xin(7:0), CLK, RST and output is IDCT-2d(7:0)

Internal Diagram:

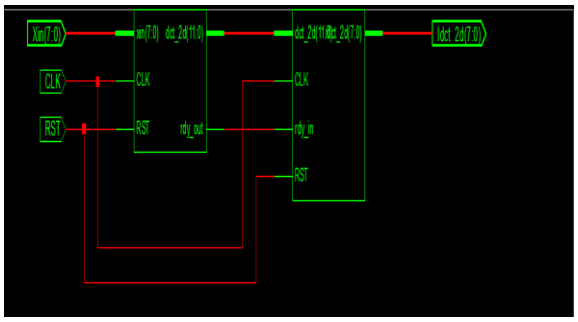


Fig.4.2 Internal diagram for DCT-IDCT

Internal diagram of DCT-IDCT consists of two modules DCT and IDCT. So DCT module have the inputs Xin, CLK, RST and outputs are IDCT-2d,Rdy-out. This DCT module is interfaced to IDCT module. It consists DCT-2d, CLK, RST, Rdy-in.

Synthesis Report:

A brief summary of FPGA resource utilization as given by Device Utilization Summary in Xilinx ISE is given in Table.

I present the synthesis results in bar graphs so that one can easily compare the performances of different FPGA technologies implementing DCT/IDCT.

Xilinx's FPGA processes the highest number of SIF frames per second then other FPGA. The other two FPGA are also processing frames higher than the minimum required rates. In fact our implementation of DCT/IDCT in hardware shows high SIF frame processing rates especially in case of using Xilinx's FPGA.

it is evident number of CCIR-TV frames processed per second by Xilinx's FPGA are much higher. The other two FPGA are also processing frames higher than the minimum required rates. Hence, our implementation of a fast DCT/IDCT shows the advantage in terms of higher frame processing rates.

Timing Report:

==========

* Final Report *

==========

Final Results

RTL Top Level Output File Name : IDCT.ngr

Top Level Output File Name : IDCT

Output Format : NGC

Optimization Goal : Speed

Keep Hierarchy : NO

Design Statistics

IOs : 18

Cell Usage :

BELS : 8756

BUF : 7

IV. TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.

FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT

GENERATED AFTER PLACE-and-ROUTE.

Clock Information:

Clock Signal	Clock buffer(FF name)	Load
CLK	BUFGP	3993

Conclusion:

In this thesis, By going this project, I reported the results from an investigation on reconfigurable implementations of DFT mapped on different FPGA technologies. Synthesis and simulation results from the experiments indicate that real-time requirements of SIF, CCIR-TV and even HDTV can be met by the implemented DFT designs. From the reported results I can conclude that all investigated FPGA implementations can speed up IDCT based compression standards dramatically. However, for computationally intensive algorithms like IDCT better results can be achieved by coarser-grained reconfigurable logic, like the one realized by the VIRTEX-II FPGA.

This thesis also described the implementation of an 8x8 IDCT and for different FPGA technologies using Modified Loeffler Algorithm .For different frame formats, all the stand-alone FPGA units were able to provide processing rates higher than the minimum rates. In particular, Xilinx's FPGA was even able to process HDTV frames at twice the required rate. These results indicate the main advantages of DFT implementation to provide the required video processing performance.

Future Scope:

Most of the IDCT based compression algorithms have been implemented in software using high-level languages such as C or Java. Such an approach provides maximum flexibility but lacks performance. At the same time, proprietary hardware implementations in ASIC were proposed to speed up such compression algorithms. But this is an inflexible solution and lacks cost efficiency.

A new approach is the combination of programmable processors with reconfigurable hardware units. Non-time critical operations are implemented in software (providing flexibility) while time -critical operations such as IDCT is implemented in hardware using FPGA (providing speed up). We must note that the performance of FPGAs is quickly nearing that of ASICS. Therefore, our IDCT implementation also targeted different FPGA technologies. In the mentioned approach, reconfigurable hardware is augmented like a functional unit, commonly referred to as a reconfigurable unit. As future work. I would like to propose the possible integration of our FPGA design in such processors. In particular I also propose the integration of our design in the MOLEN processor[R03] , which utilize microcode to control both the reconfiguration and execution process of the reconfigurable unit. Additionally, research is being carried out to further reduce the I/O ports delay of FPGA to speed up computation.

REFERENCES

[1] M. Frigo and S. Johnson, "The design and implementation of fftw3," Proc. IEEE, vol. 93, no. 2, pp. 216–231, 2005.
[2] J. Keiner, S. Kunis, and D. Potts, "Using nfft 3—A software library for various nonequispaced fast Fourier transforms," ACM Trans.Math. Softw. (TOMS), vol. 36, no. 4, pp. 19–19, 2005.
[3] D. Sepiashvili, "PerformanceModels and SearchMethods for Optimal FFT Implementations," M.Sc. thesis, Carnegie Mellon Univ., Pittsburgh, PA, 2000.
[4] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput, vol. 19, no. 90, pp. 297–301, 1965.
[5] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," Electron. Lett., vol. 20, pp. 14–16, Jan. 1984.
[6] I. Kamar and Y. Elcherif, "Conjugate pair fast Fourier transform," Electron. Lett., vol. 25, no. 5, pp. 324–325, Apr. 1989.
[7] S. Bouguezel, M. Ahmad, and M. Swamy, "A new radix-2/8 FFT al-gorithm for length- DFTs," IEEE Trans. Circuits Syst. I, vol. 51, no. 9, pp. 1723–1732, Sep. 2004.
[8] E. Dubois and A. Venetsanopoulos, "A new algorithm for the radix-3 FFT," IEEE Trans. Acoust., Speech, Signal Process,, vol. ASSP-26, pp. 222–225, Jun. 1978.
[9] S. Prakash and V. Rao, "A new radix-6 FFT algorithm," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-29, no. 4, pp. 939–941, Aug. 1981.
[10] Y. Suzuki, T. Sone, and K. Kido, "A new FFT algorithm of radix 3, 6, and 12," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-34, no. 2, pp. 380–383, Apr. 1986.
[11] D. Takahashi, "A new radix-6 FFT algorithm suitable for multiply-add instruction," in Proc. ICASSP'00, 2000, vol. 6, pp. 3343–3346, IEEE.
[12] R. Stasinski, "Radix-k FFT's using k-point convolutions," IEEE Trans. Signal Process., vol. 42, no. 4, pp. 743–750, 1994.
[13] G. Bi and Y. Chen, "Fast DFT algorithms for length ," IEEE Trans. Circuits Syst. II: Analog Dig. Signal Process. IEEE Trans-actions on, vol. 45, no. 6, pp. 685–690, 1998.
[14] M. Vetterli and P. Duhamel, "Split-radix algorithms for length- DFT's," IEEE Trans. Acoust., Speech, Signal Process., vol. 37, no. 1, pp. 57–64, 1989.

