FULL CUSTOM VLSI IMPLEMENTATION OF HIGH-SPEED 2-D DCT/IDCT CHIP

Vishnu Srinivasan† *and K. J. Ray Liu*

Electrical Engineering Department
Institute for System Research
University of Maryland at College Park
College Park, Maryland 20742

ABSTRACT

In this paper we present a full-custom VLSI design of high-speed 2-D DCT/IDCT processor based on the new class of time-recursive algorithms and architectures which has never been implemented to demonstrate its performance. We show that the VLSI implementation of this class of DCT/IDCT algorithms can easily meet the high-speed requirements of HDTV due to its modularity, regularity, local connectivity, and scalability. Our design of the 8 × 8 DCT/IDCT can operate at 50 MHz with a 400 Mbps throughput based on a very conservative estimate under 1.2μ CMOS technology.

1. INTRODUCTION

Recent advances in various aspects of digital technology have brought about many applications of digital video such as HDTV, teleconferencing, and multimedia communications. These applications require high-speed transmission of vast amounts of video data. Most video standards such as HDTV video coding, H.261, JPEG, and MPEG use DCT as a standard technique. DCT is however very computationally intensive. To realize high-speed and cost-effective DCT for video coding, one needs efficient VLSI implementations so that the high throughput requirements can be matched. There has been considerable research in efficient mapping of these algorithms to practical and feasable VLSI implementations in the recent past [1, 2]. These have however employed irregular butterfly structures with global communications resulting in complex layout, timing, and reliability concerns which severely limit the operating speed and expandability in VLSI implementations.

In this paper we present a novel VLSI implementation for the computation of the 2-D DCT/IDCT. The time-recursive architectures are developed in [3, 4]. The complexity of this class of parallel architectures is low, e.g. only $4N-4$ multipliers are needed for computing the 2-D DCT. To perform inverse DCT (IDCT), the computational structure is the same with only an additional multiplier needed. Thus, the DCT and IDCT can be naturally combined and implemented together. This class of architectures has excellent scalability, i.e. the transform size N can be made any integer by adding or deleting computational modules [4, 3]. Being a highly parallel, modular, regular, fully-pipelined, and locally-connected structure, it is a very good candidate for high-speed video applications. Also, the architecture is suited for real-time applications as the time-recursive concept has been exploited to eliminate the waiting time for data to arrive. From the VLSI implementation point of view, as these parallel IIR structures are decoupled into independent modules, the need for global communication is eliminated. The chip design has been carefully optimized based on appropriate choice of wordlength and device elements to meet the expected signal-to-noise ratio, the design of distributed arithmetic ROM units, and transformation and re-distribution of clocking and pipelined stages to improve the throughput. Timing simulations of the 8 × 8 2-D DCT/IDCT chip shows that it can easily operate at a system clock rate of 50 MHz with 400 Mbps throughput under 1.2 μ CMOS technology, which implies that it can perform DCT/IDCT under the HDTV requirements.

2. ALGORITHM & ARCHITECTURE

The IIR algorithm for the computation of the DCT is a direct 2-D method and does not require transposition, unlike most traditional row-column algorithms. The IIR structure is derived by considering the transform operation to be a filter which transforms the serial input data into their transform coefficients. The transfer function for the N-block 1-D forward and inverse transform can be shown to be:

$$H_c(z) = (-1)^k C(k)\sqrt{\frac{2}{N}}\cos\left(\frac{\pi k}{2N}\right)\frac{(1-z^{-1})}{1-2\cos\left(\frac{\pi k}{N}\right)z^{-1}+z^{-2}} \quad (1)$$

$$H_{ic}(z) = \frac{\sqrt{\frac{2}{N}}\cos\frac{(2n+1)(N-1)\pi}{2N}}{1-2\cos\left(\frac{(2n+1)\pi}{2N}\right)z^{-1}+z^{-2}} + \sqrt{\frac{2}{N}}\left(\frac{1}{\sqrt{2}}-1\right)z^{-(N-1)} \quad (2)$$

Eq. 1 and Eq. 2 suggest that both DCT and IDCT share almost the same computational structure, i.e. both the forward and inverse transform can be computed in a single architecture with minor modifications as indicated by dashed lines in Fig. 1.

The kernel shown in Fig. 1 computes a single DCT channel coefficient, based on the multiplier coefficient encoded in

This work was supported in part by NSF grant MIP9309506, ONR grants N00014-93-1-0566 and N00014-93-11028, and Maryland Industrial Partnership MIPS/Micro-Star grant.

† Vishnu Srinivasan is currently with Crystal Semiconductor Corporation, P.O. Box 17847, Austin, TX 74760.

0-8186-6950-0/94 $4.00 © 1994 IEEE

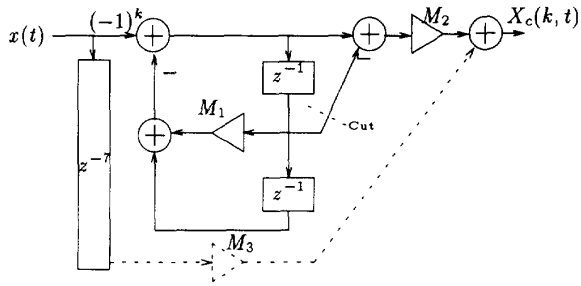


Figure 1: IIR Structure for DCT/IDCT computation

that particular filter. N such parallel modules (each with the appropriate multiplier coefficient corresponding to k) form a filter bank which computes the N coefficients of the 1-D transform. Every N cycles, the 1-D transform coefficients for a new data set is computed in parallel by the N filter bank modules. These 1-D transform coefficients are then fed into an identical but slowed down (N times) filter bank, which computes the 2-D transform of the N^2 data block.

3. FINITE WORDLENGTH CONSIDERATIONS

The implementation of the 2-D DCT/IDCT algorithm with finite precision arithmetic introduces truncation errors. To minimize the effect of truncation errors, one needs to increase the register length i.e. have a larger internal bus precision. Doing so results not only in larger area, but also affects the speed of submodules such as adders and multipliers. So we need to choose the optimum register length, which while ensuring the minimum accuracy criteria, would also lead to a high-speed implementation with small chip area.

The truncation noise introduced in the system is quantified by Peak [1] and Average SNR. Truncation noise depends upon the various system parameters like internal bus precision, and input word precision of the ROM lookup table. Architectural simulations are performed in C to model the truncation noise of the IIR structure for various system parameter choices. The simulations model the truncation behaviour when a ROM lookup table is used to implement multiplication. We have also considered using a parallel multiplier. Based on speed and area considerations, the ROM lookup was finally chosen for implementation. Table 3 compares the various ROM lookup tables and parallel multiplier. Timing simulations using 2.0μ technology parameters indicated that the ROM lookup (15ns) was four times faster than the parallel multiplier (80ns).

For many video standards we need to ensure a minimum PSNR of 40 dB. Table 3 highlights the architectural simulations of the IIR structure for the LENA test image. 4096 blocks of 8×8 pixels are used to collect the SNR statistics. Our simulation results indicate that to meet the accuracy criterion, it is sufficient to use the ROM with 12-bit input and a 16-bit internal bus precision.

Table 1: ROM vs Multiplier Comparisons

	Precision	Size
ROM (one coeff.)	12 x 12	1217λ x 1532λ
ROM	12 x 12	1292λ x 1646λ
ROM	12 x 16	1292λ x 1854λ
ROM	16 x 16	Not feasable
Multiplier	16 x 16	3513λ x 1568λ

Table 2: Finite precision simulation of IIR structure.

# of ROM input bits	Internal bus precision	Average SNR	Peak SNR
12	12	21.1 dB	27.3 dB
12	16	37.8 dB	44.0 dB
16	16	38.0 dB	44.2 dB

4. VLSI DESIGN AND IMPLEMENTATION

The regularity, modularity and local interconnection property of this architecture lends itself to efficient VLSI implementations. To this end—of achieving a high-speed design with minimum area—a full custom approach is employed. All submodules have been designed with careful regard to area and speed issues. Particular design care is taken to ensure that the critical path modules such as ROM lookup and adder are optimized. A highly hierarchical and modular strategy is employed in the chip design. By employing such a design strategy, we not only reduce the design time and effort but also have improved reliability.

4.1. Design and Simulation Tools

The VLSI layout editor MAGIC is used for implementing the full-custom DCT/IDCT chip. The various submodules needed to implement the SFG shown in Fig. 1—ROM lookup, adder, latch, delay, multiplexer, and invertor—are laid out first. These modules are characterized and their functionality verified before they are used as macro-cells. These macro-cells are instantiated and used in the higher-level hierarchies like "1-D Channel" and "2-D Channel" modules.

Crystal, and primarily, Spice are used to perform the timing simulations. The semi-interactive tool, Crystal helps in identifying the critical path in a submodule. Spice is then used for performing more accurate timing simulations of the critical path module. In our design, the ROM lookup and the adder turn out to be the critical path modules. Functionality is verified using IRSIM which is an event-driven logic-level simulator. IRSIM is used to perform logic-level simulations and verify the functionality at all hierarchy levels—starting from the bottommost, like that of the macrocells, to intermediate levels such as 1-D/2-D channel modules, then to the topmost level, namely the entire 2-D DCT/IDCT chip.

4.2. Distributed Arithmetic

This is perhaps one of the most critical sub-modules designed in the project. As this block is arrayed 32 times,

it takes up a significant chunk of the chip-area. For this reason, sufficient care has been taken in optimizing its area. Our accuracy simulations indicate the need for a ROM with 16-bit data and 12-bit address bus. A straightforward implementation of this ROM would need 2^{12} (or 4096) rows of 16-bit words.

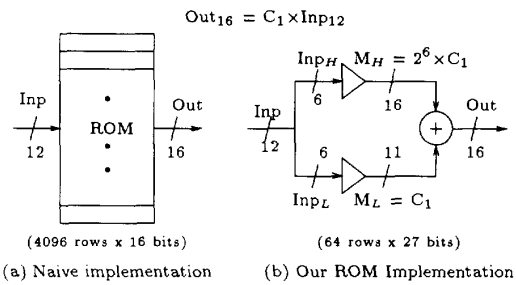


Figure 2: ROM Design Strategy

A better approach would be to split up the 12-bit input into two 6-bit words, Inp_L and Inp_H. The multiplication is effected in the following manner. The output, $Out = C_1 \times Inp$, is computed as:

$$Inp = 2^6 \times Inp_H + Inp_L.$$

$$Out = 2^6 \times C_1 \times Inp_H + C_1 \times Inp_L.$$

The two sub-products are precomputed with sufficient accuracy and storing in the ROM lookup table. The output is formed by adding the sign-extended lower order product to the higher order product. This is illustrated in Fig. 2 (b). We now need only two tables, each with 2^6 or 64 row. The main components of the ROM are tree-based row decoder, memory cells, and sense-amp.

6-64 Decoder: The 6-bit input lines are decoded and the appropriate ROM row select line is selected. Instead of a straightforward 6-bit decoder implementation, we use two 3-bit decoders and an array of 64 AND gates. This technique helps reduce the layout complexity and also results in shorter access time.

Memory cells: To minimize the area, we have used only N-type devices. A 0 or 1 is encoded by a N-type transistor connected between the output and VDD or GND respectively. The transistor gate is tied to the row-select line of the decoder. In our ROM table there are totally $(16 + 11) \times 2 \times 64$ or 3456 unit cells. Each cell corresponds to one bit of storage. The pitch of these cells is designed to be half that of the sense-amp (SA) which lets up place two cells for a single SA.

Sense-Amp: The function of this simple module is to speed up the word-lookup. It works on the simple principle of precharging the bit-lines to an intermediate voltage between VDD and GND. In this way, regardless of whether the bit-lines are turned on or off, the delay time is reduced. In the precharge phase of the clock, a p-MOS shorts the input and output of the invertor, which forces the bit-lines to the invertor transition point of about 2.5 volts. In the evaluate phase, the p-MOS transistor turns off and the bit-line signal is latched at the output through the simple butterfly-pass transistor. The SA measures $26\lambda \times 94\lambda$. The pitch of the SA is twice that of the unit-cell, allowing two memory cells for every SA. By placing one set of SA at the bottom, and another at the top, the product for two coefficients is computed at the same time.

Implementation: The first ROM is designed in the conventional manner—layout the individual cells like sense-amp, 3-bit decoders, 6-bit decoders, and the 0/1 memory cells in their various orientations at up/down locations. Once the complete ROM is assembled, the location and arraying information of each cell is noted. A perl script is used to assemble new ROMs with different coefficients using the above information. The size of the basic ROM structure (with SA) which encodes two multiplier coefficients is $1292\lambda \times 1854\lambda$. If we include the adders to combine the high and low order products, the ROM/adder assembly measures $2226\lambda \times 1854\lambda$.

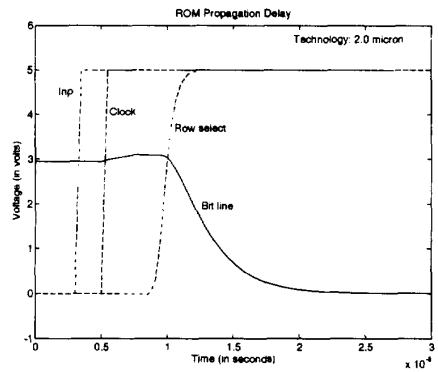


Figure 3: 2.0μ ROM Timing Simulation

Timing issues: Spice is used to estimate the propagation delay of the ROM lookup. Layout parasitics are extracted for 2.0μ technology scaling parameters. From Fig. 3 we see that the propagation delay is about 14ns. Simulations using 1.2μ parameters was less than 9ns.

4.3. Adder

There are several possible designs for adders. Implementations can employ either the simple, but slow ripple carry adder or utilize the fast, but complex carry lookahead (CLA) adder. In our case, accuracy simulations indicated the need for a 16-bit wide internal bus. Preliminary analysis revealed that a simple ripple adder woud have been too slow for our timing demands. At the same time building a CLA scheme for 16-bits is complex. A good compromise was to implement a 4-bit CLA, and connect up four of these units in a ripple fashion to obtain a 16-bit carry lookahead-cum-ripple adder. The basic 4-bit carry lookahead adder implementation is based on [5]. This module has 704 transistors and measures $1348\lambda \times 355\lambda$ units.

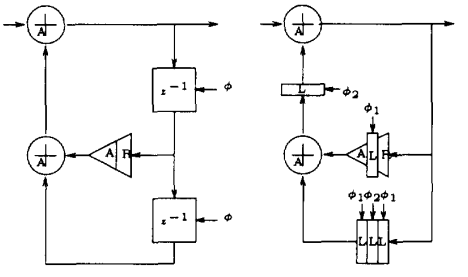
Timing: The longest delay in the adder is caused by the signal propagated from LSB or carry-in to the MSB. In our case, 2.0μ and 1.2μ simulations indicated proagation delays of 19ns and 9ns respectively.

4.4. Other Macro-blocks

16-bit wide dynamic half-latches with reset, shift registers with 1/7/8 clock-cycle delays, 2-input, 16-bit multiplexers, and clock-buffers are some of the other modules which designed and laid out. Typically, the bit-slice is designed first. It is then arrayed to form the required bus width. Functionality is verified, and output transistors are sized to provide adequate drive-capability. Some of the block sizes are as follows: 'delay-1' module measures $938\lambda \times 217\lambda$, the 'delay7' measures $1028\lambda \times 1549\lambda$, the 'delay8' measures $1028\lambda \times 1771\lambda$, and the multiplexer measures $1271\lambda \times 119\lambda$.

4.5. Clocking

The propagation delays of ROM and adder helps us decide retiming and clocking issues. Fig. 4 illustrates the implementation of the 1-D kernel SFG using a single-phase clock (with static latches) or a two-phase non-overlapping clock (with dynamic latches). From both speed and area viewpoint the two-phase clocking scheme turns out to be a better choice.



(a) Single phase clock (b) Two-phase clocking scheme

Figure 4: Clock Speedup

A two-phase clock permits us to use dynamic latches which are simpler to design and are more compact when compared with static latches. Two-phase clocks gives us the flexibility to retime the SFG such that the delays in the various critical paths are equalized. This is illustrated in Fig. 4 (a) and (b). In Fig. 4 (a), the propagation delay between any two subsequent latches is (3A+R, 2A), where A and R are the propagation delays of adder and ROM. The maximum delay between two subsequent latches is the critical path and will determine the fastest possible clock rate. In Fig. 4 (a) it is (3A+R). Knowing that the adder and ROM delay are about the same, the critical path is retimed as shown in Fig. 4 (b). The maximum delay now is either (A+R) or 2A; which means that the critical path delay is halved, and thus the maximum clocking rate is doubled.

4.6. Higher Hierarchy Levels

The 1-D kernel shown in Fig. 1 is implemented using the various macro-cells which have been previosly designed. Fig. 5 illustrates the VLSI implementation of the SFG. Depending on the value stored in the ROM, this particular module computes the appropriate DCT/IDCT transform coefficient.

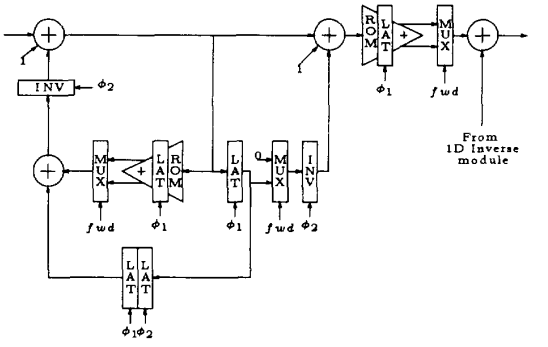


Figure 5: 1-D IIR SFG with clocks and latches

The physical layout of the 1-D channel is shown in Fig. 6. All of the eight channel modules are identical except that they instantiate different ROM tables. The circuit has been laid out in such a manner that it facilitates easy modular development. Inter-module connections are brought to the edges of the blocks where they get connected with the other modules wire-segments when these modules are tiled. By adopting such a methodology, we save considerably in design time and effort, and at the same time, if the modules are designed properly (matching pitch), we save in interconnection area requirement also. The power rails, input/output, and other important control signals are routed from top to bottom in each module. As they are tiled vertically the routing of all signals is done automatically. We only have to concern ourselves with feeding the input signals to the entire 1-D module, either from the top or the bottom, as local distribution of these signals is already taken care of. The 1-D kernel measures $2742\lambda \times 8029\lambda$. The eight 1-D channel modules are tiled one over the other to form the complete 1-D DCT engine.

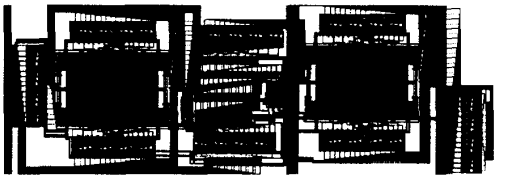


Figure 6: VLSI layout of 1-D channel.

The 2-D Channel Module is also designed in a similar manner. The SFG is almost the same, except that it is slowed down N times—a single delay unit is replaced by N delay units. This facilitates computation of 8 blocks of data in a time-displaced parallel fashion. The 2-D kernel measures $2784\lambda \times 10672\lambda$.

The Circular Shift Array (CSA) was the last module to be designed. It takes the 8 parallel 16-bit words generated by the 1-D module and feeds it serially to the 2-D module.

The CSA module serves another important function; that of storing the 1-D IDCT coefficients of the first row required for computation of the inverse at the second stage. There are several control signals for this module—to read data from the 1-D module, to latch it in, and shift it out serially to the 2-D module, to latch in data to help in computing of the inverse, and to hold it until required.

4.7. Miscellaneous Issues

Routing of the control signals is a fairly important issue as there are quite a few control signals that need to be distributed to various modules like latches, delays, and multiplexers. The basic idea is to distribute the master control signals to higher-level cells like 1-D/2-D channel modules and use a local buffer to generate the control signals, which are then distributed to all submodules within that block. This is essentially a multi-level tree distribution of the control signals. By employing such a scheme, we are not only able to minimize skew, but also have improved rise and fall times. This scheme is particularly relevant to the clock signal distribution.

The 'rst' control signal is to be distributed to all those modules that are clocked as they need to be reset between blocks. The 'fwd' signal which determines whether the forward or inverse DCT is computed is routed to all the multiplexers. Modules require these control signals also need the complement, which is generated at the local buffer. It is not necessary to route the complement of the control signals on a global chip scale.

4.8. 2-D DCT/IDCT Chip

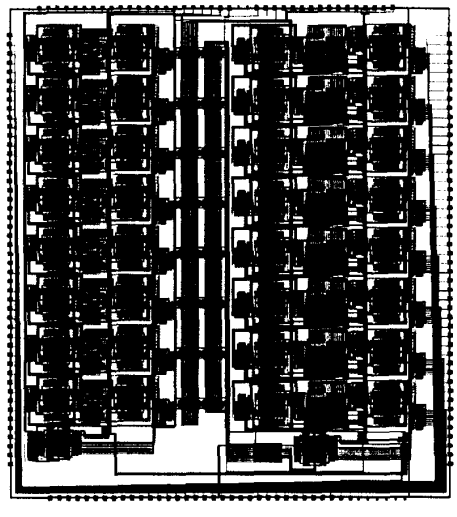


Figure 7: Two Dimensional DCT/IDCT Chip. Chip measures $24550\lambda \times 27094\lambda$. Area is 240 mm^2.

Using all the cells—1-D, 2-D, and CSA—already described, the final chip is assembled. The physical layout of the 2-D DCT/IDCT chip is shown in Fig. 7. The chip statistics are tabulated in Table 3. Fabricated in 1.2μ CMOS N-well technology, this chip has a transistor density of 1.3 KT/mm^2. Spice simulations performed using 1.2μ technology parameters correspond to a critical path propagation delay of less than 20ns. The active pin count of the chip is only 38. Several internal pins helpful for debugging purposes are brought out.

Table 3: 2D-DCT/IDCT Chip Statistics

Technology	1.2μ CMOS N-well
Die Dimensions	$24550\lambda \times 27094\lambda$
Chip Area	240 mm^2
# of Transistors	320,000
Speed	50 MHz
Data rate	400 Mb/s

5. CONCLUSIONS

In this paper, we have presented a VLSI implementation of a high-performance high-speed 2-D DCT/IDCT chip. It is a full-custom implementation employing a highly modular and hierarchical design strategy. Distributed arithmetic is used for fast and compact multipliers. Non-overlapping two-phase clocking scheme leads to a faster and more compact layout of the kernel. Architectural simulations are conducted for choosing system parameters that ensure adequate accuracy while minimizing chip area. The chip dimensions are $24550\lambda \times 27094\lambda$ and its area is 240 mm^2 based on 1.2 μ technology. The pin-count is 176, and the chip has over 320,000 transistors. Timing simulations performed using Spice indicate a clock frequency of 50 MHz corresponding to a data throughput of 400 Mb/s. We have shown that VLSI design based on the class of time-recursive algorithms and architectures can easily meet the high-speed requirements necessary for video coding applications. This chip has been submitted for fabrication in 1.2μ CMOS N-well double-metal single-poly technology.

6. REFERENCES

[1] M.-T. Sun, T.-C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16x16 Discrete Cosine Transform," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 610–617, April 1989.

[2] T. Miyazaki, T. Nishitani, M. Edahiro, I. Ono, and K. Mitsuhashi, "DCT/IDCT Processor for HDTV developed with DSP Silicon Compiler," *Journal of VLSI Signal Processing*, no. 5, pp. 151–158, 1993.

[3] C. T. Chiu and K. J. R. Liu, "Real-time parallel and fully pipelined two-dimensional DCT lattice structures with applications to HDTV systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 25–37, March 1992.

[4] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive Discrete Cosine/Sine/Hartley transforms," *IEEE Trans. Signal Processing*, vol. 41, pp. 1357–1377, March 1993.

[5] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*. Addison-Wesley, 1988.