VLSI Design of High-Speed Time-Recursive 2-D DCT/IDCT Processor for Video Applications

by V. Srinivasan and K.J.R. Liu

T.R. 94-60

ISR
INSTITUTE FOR SYSTEMS RESEARCH

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **1994**	2. REPORT TYPE	3. DATES COVERED **00-00-1994 to 00-00-1994**

4. TITLE AND SUBTITLE
VLSI Design of High-Speed Time-Recursive 2-D DCT/IDCT Processor for Video Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical Engineering,Institute for Systems Research,University of Maryland,College Park,MD,20742

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		**33**	
unclassified	**unclassified**	**unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

VLSI Design of High-Speed Time-Recursive 2-D DCT/IDCT Processor for Video Applications [†]

Vishnu Srinivasan and K. J. Ray Liu

Electrical Engineering Department and 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 prove 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. In comparison to the existing designs, our approach offers many advantages that can be further explored for even higher performance.

[†] 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.

1 Introduction

Recent advances in various aspects of digital technology have made possible 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 discrete cosine transform (DCT) as a standard transform coding scheme [1, 2, 3, 4]. The 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 [5, 6, 7, 8, 9]. 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.

Recently a new class of transform coding architectures based on time-recursive approach has been proposed [10, 11, 12]. 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 [12]. 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 [10, 11, 12]. In addition, these are highly parallel, modular, regular, fully-pipelined, and locally-connected. Thus it is a very good candidate for high-speed video applications. Also, the architecture is very suitable 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 the parallel computational IIR structures are decoupled into independent modules, the need for global communication is eliminated.

In this paper we present a novel VLSI implementation for the time-recursive 2-D DCT/IDCT processor. The class of time-recursive parallel architectures has never been designed and implemented to prove its superior properties— our goal here, is to show its performance under full-custom VLSI implementation and to make comparison with other existing VLSI designs based on different algorithms. 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. The simulation of our design of the 8×8 2-D DCT/IDCT 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.

The paper is organized in the following manner. We give a brief summary of the algorithm and architecture in Section 2. The finite wordlength and architectural considerations are given in Section 3. The VLSI design and implementation are detailed in Section 4. A comparison to existing DCT/IDCT VLSI design is presented in Section 5, followed by the conclusion in Section 6.

2 Algorithm and Architecture

The time-recursive two-dimensional DCT for a $N \times N$ image block $\{x(m,n) : m = t, t+1, \ldots, t+N-1; n = 0, 1, \ldots, N-1\}$ is defined as [11, 12]:

$$X_c(k,l,t) = \frac{2}{N} C(k)C(l) \sum_{m=t}^{t+N-1} \sum_{n=0}^{N-1} x(m,n) \cos \frac{\pi[2(m-t)+1]k}{2N} \cos \frac{\pi(2n+1)l}{2N} \tag{1}$$

where

$$C(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0, \\ 1 & \text{otherwise.} \end{cases}$$

The time index t in $X_c(k,t)$ denotes that the transform starts from $x(t)$. The 2-D IDCT is defined in a similar manner:

$$x_c(m,n) = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_c(k,l) C(k) C(l) \cos [\frac{\pi[2k+1]m}{2N}] \cos [\frac{\pi(2l+1)n}{2N}] \tag{2}$$

DCT/IDCT is a very computationally intensive operation. To be able to use this technique for high-throughput applications such as HDTV coding, an efficient VLSI implementation is essential. In the past several fast algorithms have been mapped onto VLSI chips, but they are not particularly well adapted for VLSI where regularity, modularity, timing, layout complexity, and area are of more concern.

The IIR algorithm [12] for the computation of the DCT is a direct 2-D method and does not require transposition, unlike more traditional row-column algorithms. The structure is derived by considering the transform operation to be a filter which transforms the serial input data into their

[†] This result is a conservative estimate

transform coefficients.

The 1-D DCT and its inverse for a N block input data, starting from $x(t)$ and ending with $x(t+N-1)$ are defined as:

$$X_c(k,t) = C(k)\sqrt{\frac{2}{N}} \sum_{n=t}^{t+N-1} x(n)\cos{[(n-t+\frac{1}{2})\frac{\pi k}{N}]}, \quad k = 0,1,\ldots,N-1, \tag{3}$$

$$x_c(n,t) = \sqrt{\frac{2}{N}} \sum_{k=t}^{t+N-1} C(k-t)X_c(k)\cos{[(n+\frac{1}{2})\frac{(k-t)\pi}{N}]}, \quad n = 0,1,\ldots,N-1, \tag{4}$$

where $C(k)$ is as defined in (1).

The transfer function for the forward and inverse 1-D DCT can be shown to be:

$$H_c(z) = \frac{Y(z)}{X(z)} = C(k)\sqrt{\frac{2}{N}}\cos{(\frac{\pi k}{2N})}\,\frac{(1-z^{-1})((-1)^k - z^{-N})}{1-2\cos{(\frac{\pi k}{N})}z^{-1}+z^{-2}} \tag{5}$$

$$H_{ic}(z) = \sqrt{\frac{2}{N}}\frac{\cos{(\frac{(2n+1)(N-1)\pi}{2N})} - \cos{(\frac{(2n+1)\pi}{2N})}z^{-N} + z^{-(N+1)}}{1-2\cos{(\frac{(2n+1)\pi}{2N})}z^{-1}+z^{-2}} + \sqrt{\frac{2}{N}}(\frac{1}{\sqrt{2}}-1)z^{-(N-1)} \tag{6}$$

If we are however interested in only the N-block transform, (5) and (6) can be simplified to:

$$H_c(z) = (-1)^k C(k)\sqrt{\frac{2}{N}}\cos{(\frac{\pi k}{2N})}\,\frac{(1-z^{-1})}{1-2\cos{(\frac{\pi k}{N})}z^{-1}+z^{-2}}, \tag{7}$$

$$H_{ic}(z) = \frac{\sqrt{\frac{2}{N}}\cos{\frac{(2n+1)(N-1)\pi}{2N}}}{1-2\cos{(\frac{(2n+1)\pi}{2N})}z^{-1}+z^{-2}} + \sqrt{\frac{2}{N}}(\frac{1}{\sqrt{2}}-1)z^{-(N-1)}. \tag{8}$$

The signal flow graph (SFG) shown in Fig. 1 implements (7) or (8) i.e. the forward and the inverse 1-D DCT depending on the multiplier coefficients and the modifications to the SFG as indicated by the dashed lines.

The kernel shown in Fig. 1 computes a single DCT channel coefficient based on the multiplier coefficient encoded in 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. The block diagram for the 2-D DCT/IDCT architecture [11] is outlined in Fig. 2.

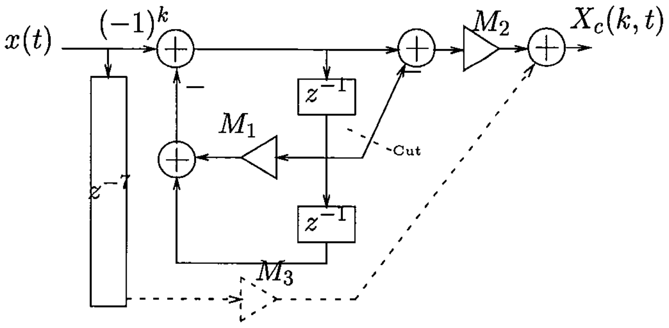


Figure 1: IIR Structure for DCT/IDCT computation

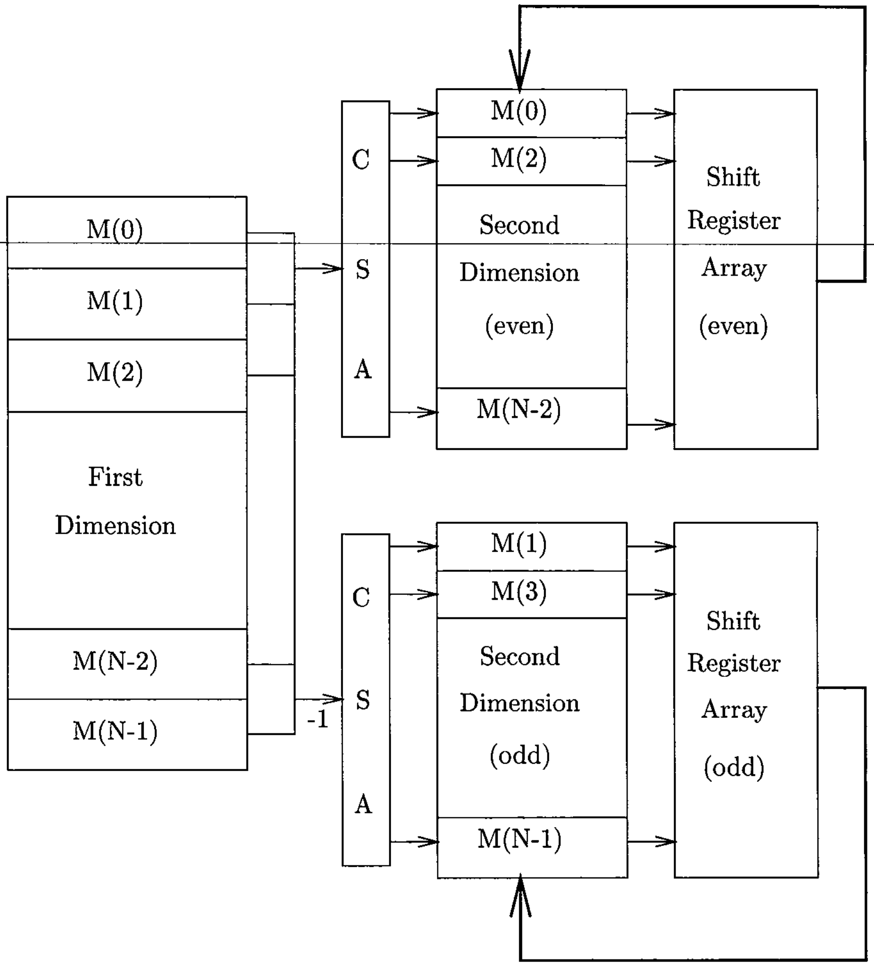


Figure 2: Block 2-D DCT/IDCT Architecture

3 Finite Wordlength and Architectural Considerations

In the realization of the DCT algorithm there are tight tradeoffs between various criteria like accuracy, speed, and area of the chip. The implementation of the 2-D DCT/IDCT algorithm with finite precision arithmetic (due to fixed register length) 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 however 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.

To make sure the accumulated errors do not exceed the video coding requirements we model the 2-D IIR DCT/IDCT architecture in C, and perform simulations to verify that Peak and Average SNR requirements suitable for video coding applications are met [5]. These simulations in conjunction with preliminary timing analysis of various submodules helped in deciding the final architecture suitable for a high-performance high-speed 2-D DCT/IDCT chip. The truncation errors which are introduced in the system are quantified by PSNR and Average SNR. The Average SNR is defined as:

$$SNR = 20 \log \frac{I(x,y)}{|O(x,y) - I(x,y)|} \tag{9}$$

where $O(x,y)$ and $I(x,y)$ are the output and input image pixel intensity values for position (x,y). The Peak SNR is defined in a similar manner with the only difference being that it focuses on the noise introduced due to truncation, and therefore assumes peak input intensity value.

3.1 ROM Lookup vs Parallel Multiplier

One of the critical submodules of this design was the multiplier implementation. There are pros and cons of using distributed arithmetic techniques [13] and parallel multipliers [14, 15]. ROM table lookups can be done very fast as compared to using a regular multiplier. Also its accuracy is higher than regular multipliers as lookup-entries are precomputed (with 64-bit precision) and stored in the table. The only drawback is that, as a table lookup, its size grows exponentially with input bit precision. Table 1 compares some of the preliminary ROM designs with the best parallel multiplier that was available to us.

We see that, barring a need for a 16-bit wide input for the ROM, its size is smaller than the

	Precision	Size	Delay
ROM (one coeff.)	12 x 12	1217λ x 1532λ	15 nS
ROM	12 x 12	1292λ x 1646λ	15 nS
ROM	12 x 16	1292λ x 1854λ	15 nS
ROM	16 x 16	Not feasable	15 nS
Multiplier	16 x 16	3513λ x 1568λ	80 nS

Table 1: ROM vs Multiplier Comparisons

parallel multiplier. Also, our unique design of the ROM allows us to have two sets of table entries interdigitated with an area increase of only 12%. This is essential if we are to compute both the forward and inverse transforms using the same structure. Thus we see, not only is the ROM smaller than the multiplier, but it is over 4 times faster than the best parallel multiplier that was available to us. The timing simulations are for the 2.0μ technology parameters ($\lambda = 1\mu$).

3.2 Finite Wordlength Simulations

For many video standards we need to ensure a minimum PSNR of 40 dB [16]. Fig. 3 illustrates the architectural simulations which help in quantifying the truncation noise in the system by PSNR computation. The blocks for forward and inverse DCT are C architectural models which model the finite precision arithmetic of the IIR structure for the 2-D DCT/IDCT computation, and helps in estimating the Peak and average SNR. Simulations are performed for a set of different system parameters aimed at meeting minimum SNR criterion while at the same time, minimizing area and maximizing speed.

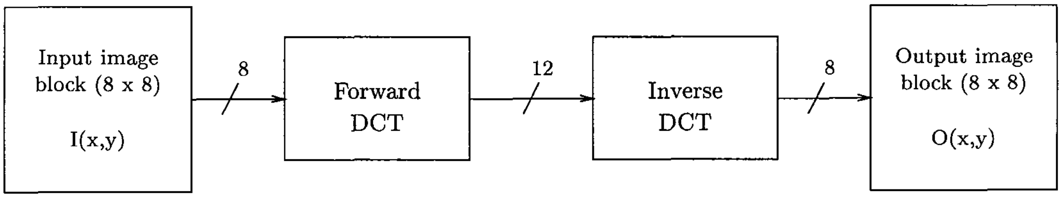


Figure 3: Accuracy simulation block-outline

Several input images—LENA, SAILBOAT, AIRPLANE, and random data—have been used in these simulations. These images are 512 × 512 pixels, with each pixel being specified by 8-bit word corresponding to 256 gray levels. As our image block size is 8 × 8 pixels, a single image yields 4096 blocks for which the 2-D DCT–IDCT is computed and the PSNR statistics collected. The highlights of these simulations are presented in Table 2.

# 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

Table 2: Finite precision arithmetic simulation of IIR structure.

As we can see, the minimum PSNR requirements are adequately satisfied if we use a system bus precision of 16-bits with a 12-bit wide ROM input wordlength. The 16-bit input, however, increase the ROM area by four times with almost the same PSNR.

4 VLSI Design and Implementation

~~The regularity, modularity and local interconnection property of this architecture lends itself to~~ efficient VLSI implementations. To achieve a high-speed design (with minimum area) we use the full-custom approach. 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/Methodology

The VLSI layout editor MAGIC is used to implement the full-custom 2-D DCT/IDCT chip. The various submodules needed for the chip design—ROM lookup, adder, latch, delay, multiplexer, and invertor—are laid out first. These submodules 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, which in turn are larger macro-cells in the next higher level. This hierarchical approach minimizes the design effort considerably, as one needs to only consider tiling, placement and routing of various modules at the current level.

Crystal, and primarily, Spice are used to perform the timing simulations. Crystal, a semi-interactive program is used to estimate the critical path in a submodule between a specified set of input and output vectors. For a given change in the input vector, worst case times/delays are propagated to the output vectors. This gives us information about the critical paths. Using this as a starting point, Spice is used to perform a detailed timing analysis of critical paths of various macrocells. In our design, the slowest modules turned out to be the ROM lookup and carry lookahead adder.

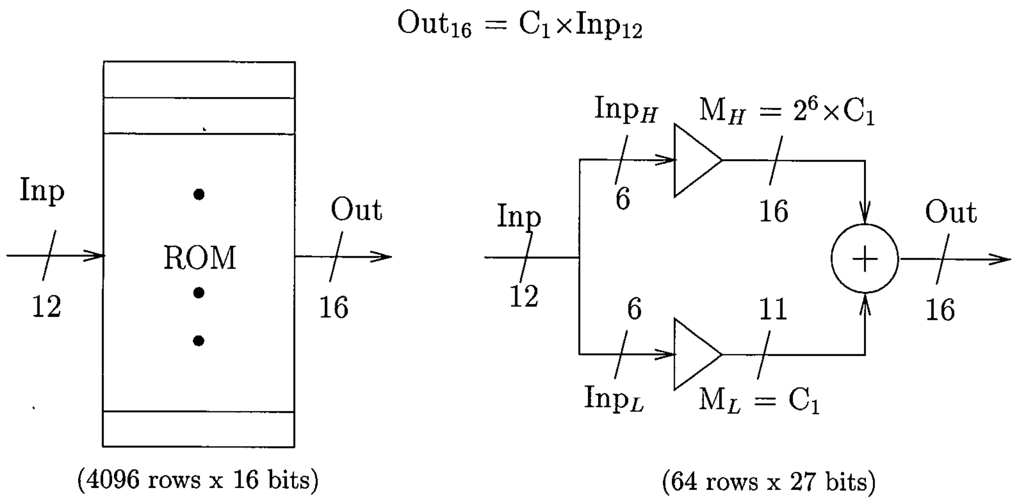
Functionality verification is done using IRSIM which is an event-driven logic-level simulator. The logic-level simulations are performed at a much higher level of circuit abstraction, treating the transistors as switches which are either ON or OFF. IRSIM is used to test not only the functionality of all macrocells, but is also 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 submodules designed in this project. The ROM is optimized both for speed and area. The area optimization is especially critical, as the ROM is arrayed 34 times in the complete chip and occupies a significant percentage of the chip area. Even a small reduction in ROM area would affect our chip area statistics considerably. While speedwise, ROMs are superior to multipliers, their area increases exponentially with the input word size.

The optimal system design, satisfying accuracy requirements and minimizing area required a 16-bit internal bus with a 12-bit ROM input wordlength. A straightforward implementation of this ROM lookup table would need 2^{12} (or 4096) rows of 16-bit words, which is too large to be implemented. However by using partial sums method, as illustrated in Fig. 4, we can reduce the size of the lookup table to 2^6 (or 64) rows, but this requires two separate lookup tables along with a fast adder to combine the high and low order sums. Also, our ROM structure should have the capability of computing the 2 products, for both forward and inverse transforms. We will see in the subsequent paragraphs how the high and low order tables and interdigitation is employed to craft a compact and fast ROM lookup.

The 12-bit input is split into two 6-bit words, Inp_L and Inp_H. The multiplication is effected in



(a) Naive implementation

(b) Our ROM Implementation

Figure 4: ROM Design Strategy

the following manner. The output, $Out = C_1 \times\ Inp$, is computed as:

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

where the input

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

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. 4 (b). We need two tables, each with 2^6 or 64 rows only. It turns out that we need to store 16 bits for the upper precomputed product and 11 bits for the lower product. The lower and higher order ROM entries are to be added with the proper shift, taking into account the 2's complement representation.

4.2.1 Design details

Most ROMs are based on the precharged scheme, where the bit lines are precharged high and during the evaluate phase, according to the stored bit-sequence, selected lines are discharged. Our ROM is based on a novel design, which reduces the access time. The main components of the ROM are tree-based row decoder, memory cells, and sense-amp. The ROM schematic is shown in Fig. 5.

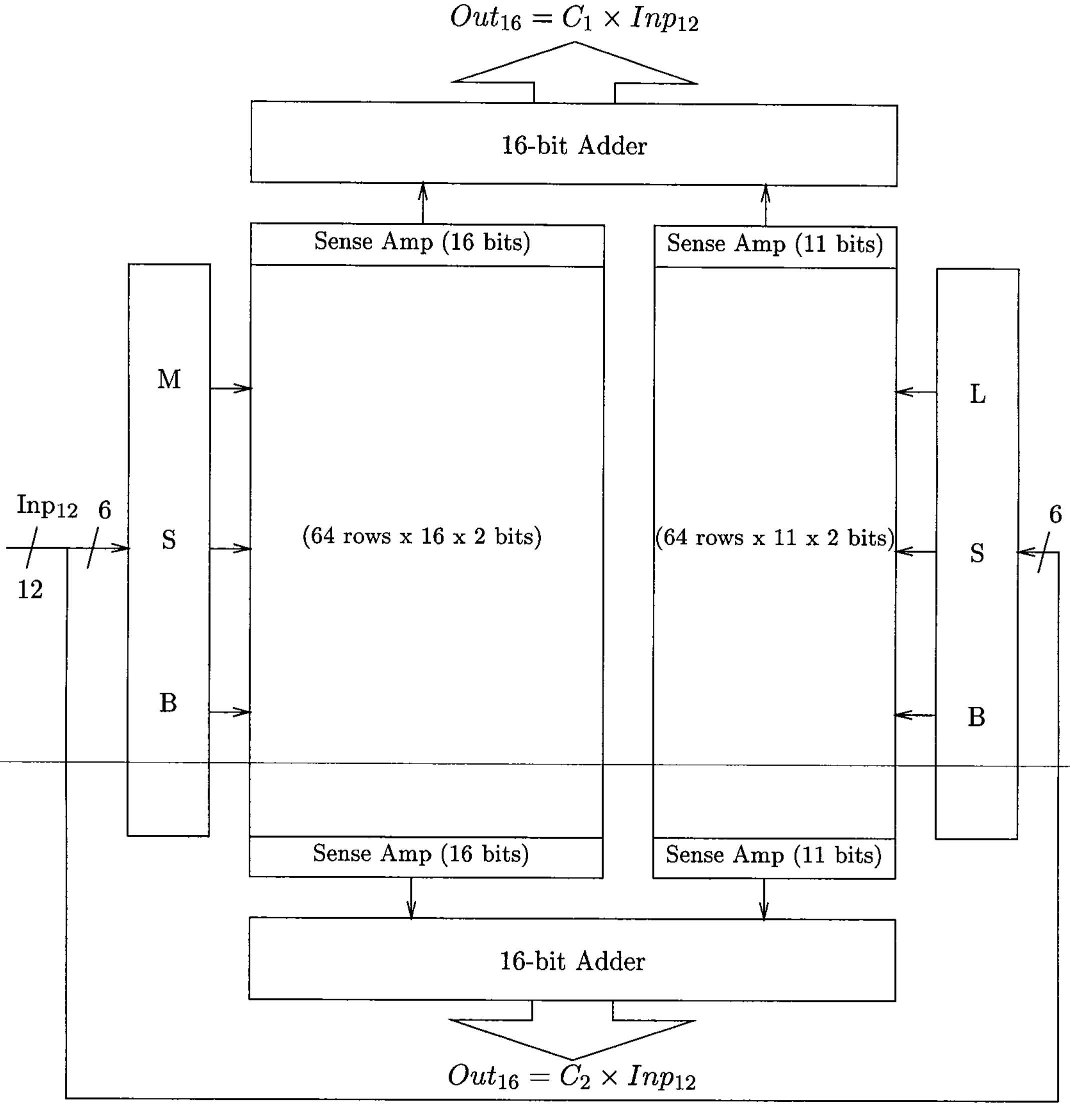


Figure 5: ROM Multiplier Schematic

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. The VLSI layout of the 6-bit decoder is shown in Fig. 6.

Lookup Table: The ROM unit cells which encode a 1 or 0 are shown in Fig. 7. These cells are identical and measure $13\lambda \times 16\lambda$. Depending on whether a logic high or low is to be stored, we have

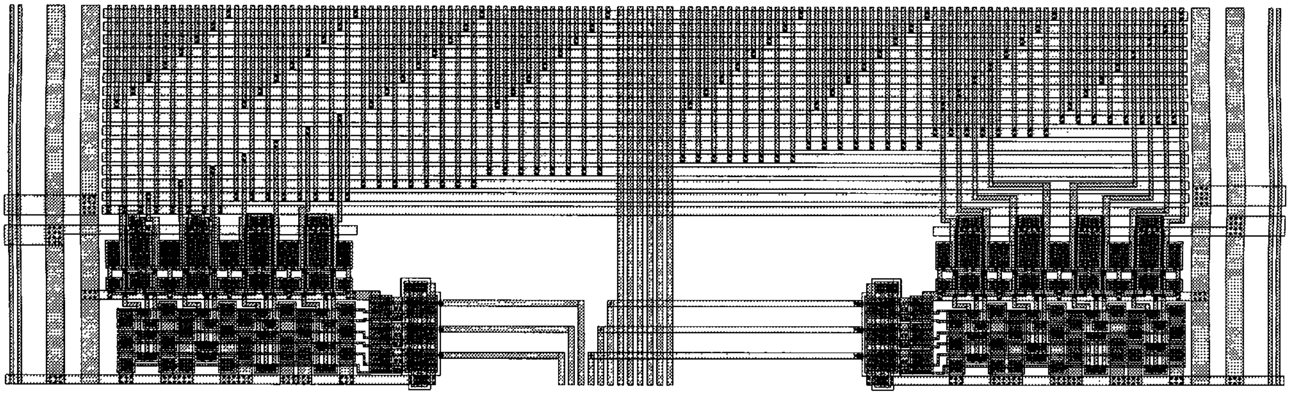


Figure 6: Physical layout 6-bit decoder ($377\lambda \times 1292\lambda$).

a n-channel transistor connected between the output line and either Vdd or GND, respectively. The transistor gate is connected 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. So by placing one set of SAs at the top of the table and another below, we are able to interdigitate two lookup tables with minimal increase (only 12%) in area requirement.

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. The sense-amp schematic is shown in Fig 8 (b). In the precharge phase of the clock (ϕ is low), the p-MOS shorts the input and output of the invertor, which forces the bit-lines to the invertor transition point at 2.5 volts (Simulations have however indicated that this voltage is closer to 3 volts). In the evaluate phase, the p-MOS transistor turns off and the bit-line signal is latched at

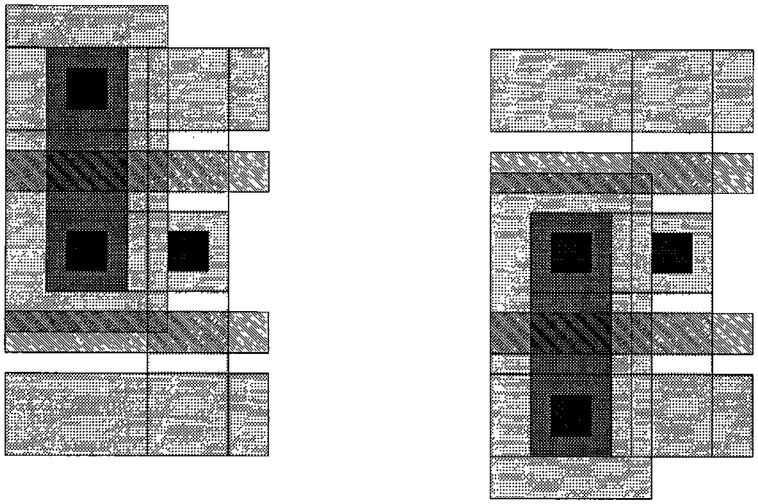


Figure 7: ROM unit-cells of size 13λ × 16λ. Encode bits 1 and 0 .

the output through the simple butterfly-pass transistor. The SA measures 26λ × 94λ. The pitch of the SA is twice that of the unit-cell, allowing two such cell 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.

4.2.2 ROM Implementation

The first ROM is designed in the regular manner—design the individual cells like sense-amp, 3-bit decoders, 6-bit decoders, and 0/1 unit cells in their various orientation and at UP/DOWN positions. Once a complete ROM is assembled (with dummy coefficients), it is broken up into subunits/cells, and their locations and arraying information noted. Using a perl script, new ROMs are assembled by stiching the various subunits/cells in the appropriate locations. The crucial part is the encoding of the interdigitated coefficients. The sine or cosine coefficients are computed using double-precision floating point arithmetic on the Sun workstation, and given as input to the perl script. These numbers are converted to 2s complement binary, and routines called to place the "1-unitcell" or "0-unitcell" depending on the bit-value at that particular location.

The perl script writes the placement information of the ROM subunits in a temporary file. The script then invokes MAGIC as a child process which does the actual job of putting together all the modules to form the final ROM submodule. The automatically generated ROM is carefully tested for functionality verification, design rule violations and timing analysis. To generate ROM lookup

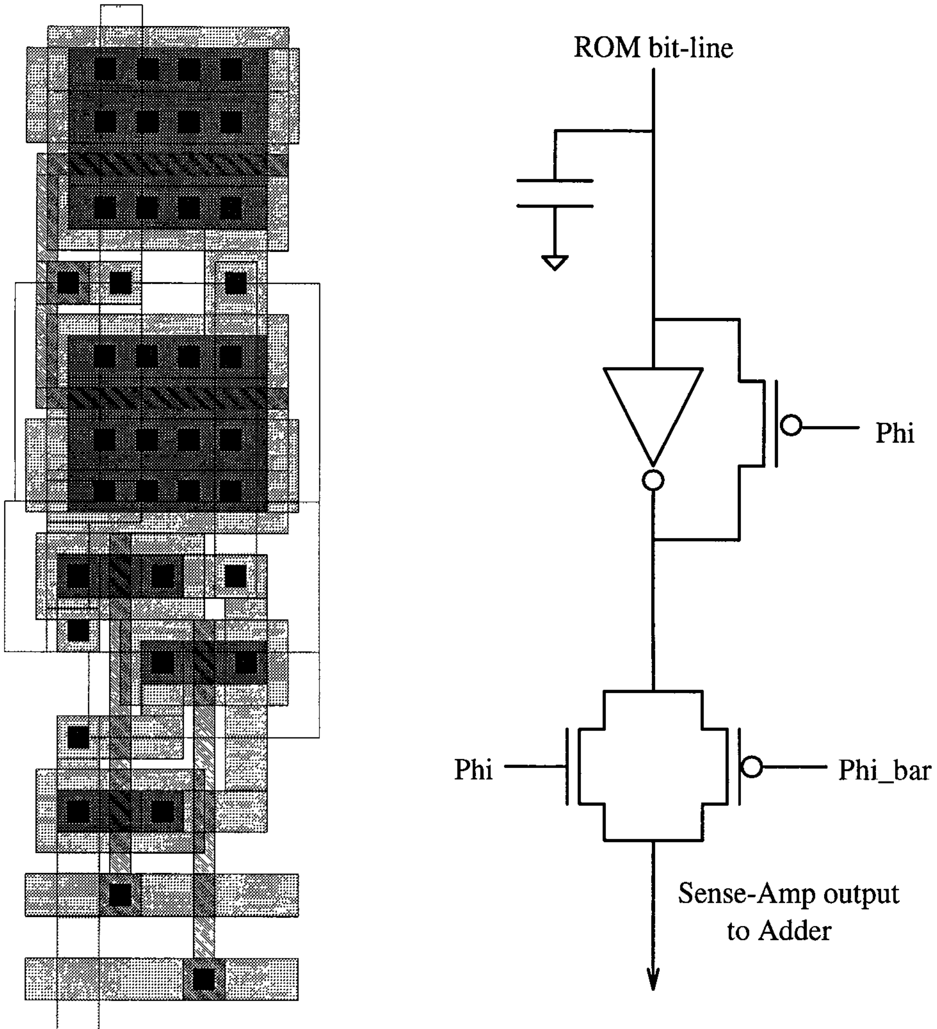


Figure 8: Sense-Amp: (a) Physical layout of SA ($26\lambda \times 94\lambda$), (b) Corresponding circuit schematic.

tables with different multipliers, the process is repeated for all the channels. Multiplier coefficients required for the various channels are computed and the ROM generator invoked for each of these.

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 $2226\lambda \times 1854\lambda$. The combined ROM lookup-table and adder to effect multiplication is shown in Fig. 9.

4.2.3 Timing

To compute the propagation delay in the ROM lookup table, Spice is used. Using extraction style for 1.2μ or 2.0μ, the circuit parasitics are extracted. The timing analysis for ROM lookup-table

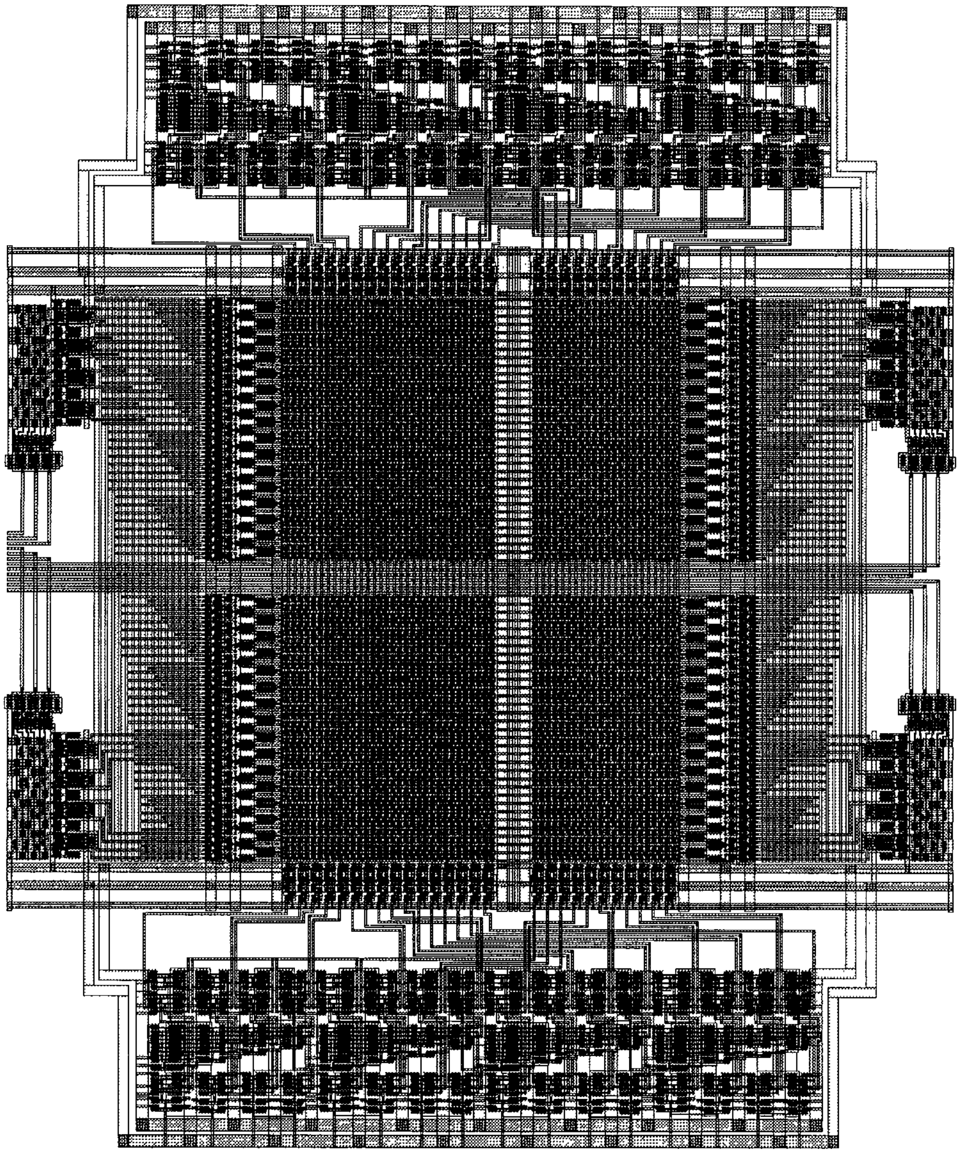


Figure 9: Physical layout of ROM lookup-table with associated adders. Dimension is 2226λ × 1854λ.

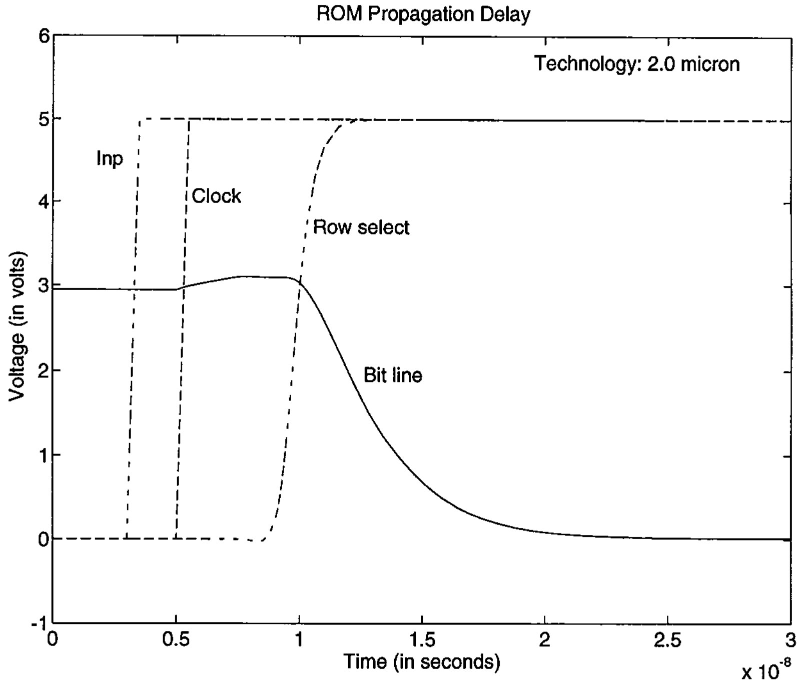


Figure 10: 2.0μ ROM

extracted using the 2.0μ technology scaling parameters (of Magic) is shown in Fig. 10. The ROM input is pulsed to 5V at 3ns and the clock at 5ns. The 'Row select' line is the output of the 6-bit decoder which selects one of the words in the ROM table. The propagation delay is defined to as the time difference between the 50% points of the input and the 'Row select' lines. The bit line is at 2.9 volts, and it discharges to 0.5 volts at 16.5 ns. The propagation delay for the ROM lookup is 13.5ns.

4.3 Adder

There are several possible designs for adders. They could be based on the simple majority logic function approach [17, 18] implemented using straightforward combinational gates. Implementation can employ either the simple, but slow ripple carry adder or utilize the fast, but complex carry lookahead (CLA) method. There are pros and cons to every choice, and depending upon the application one should choose the appropriate design.

We need to choose the design which would yield the least propagation delay, and at the same time would not require elaborate clocking precharge schemes or take up too much area. Carry

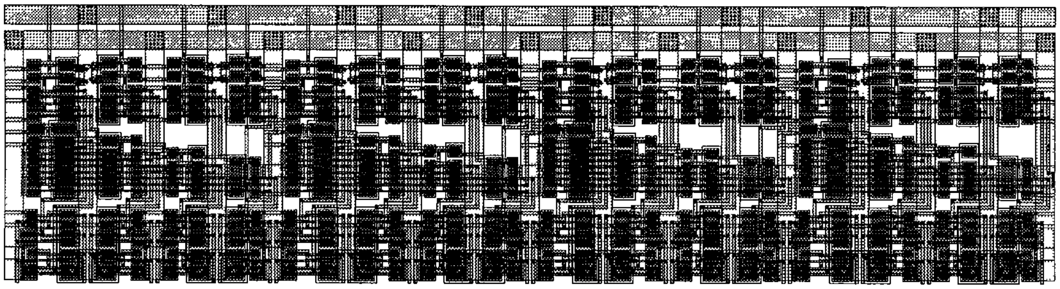


Figure 11: Physical layout of the 16-bit carry-lookahead–ripple Adder. Module dimension is 1348λ× 355λ.

lookahead adders would give us the best performance in terms of delay, but it is not practical to implement CLA with more than 4-bits as they become too large. Ripple carry adders on the other hand, would yield compact layouts, but their speed would depend on the number of bits being added.

In our case, accuracy simulations indicated the need for a 16-bit wide internal bus. Clearly, a simple ripple carry adder would be too slow for our requirements. At the same time, building a CLA scheme for 16 bits is impractical. A good compromise that we chose 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 [17]. The adder module is shown in Fig. 11. This module has 704 transistors and measures 1348λ × 355λ units.

4.3.1 Timing

We expect the longest delay in the adder to be the signal propagated from carry-in to the MSB. Here, as carry-in is always preset to 0/1, the longest delay is from the LSB to the MSB. Simulations performed using 1.2 μ technology parameters indicate a maximum propagation delay of 9 ns. The adder inputs are FFFF_{hex} and 0000_{hex}. 'InA1' is pulsed from 0 to 5 volts at 2ns. 'Out16' drops to 0V after the propagation delay as shown in Fig. 12.

4.4 Clocking

The propagation delays of ROM and adder helps us decide retiming and clocking issues. Fig. 13 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.

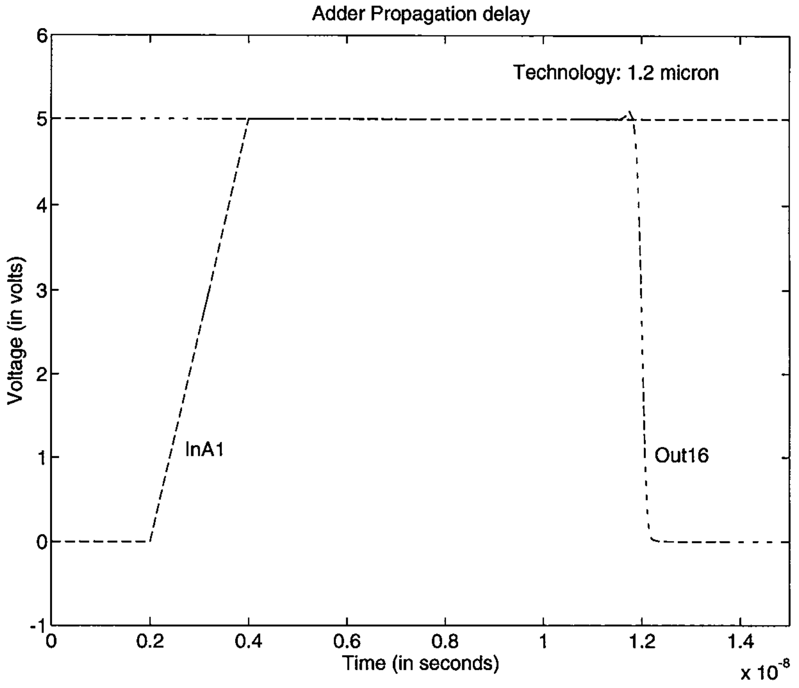
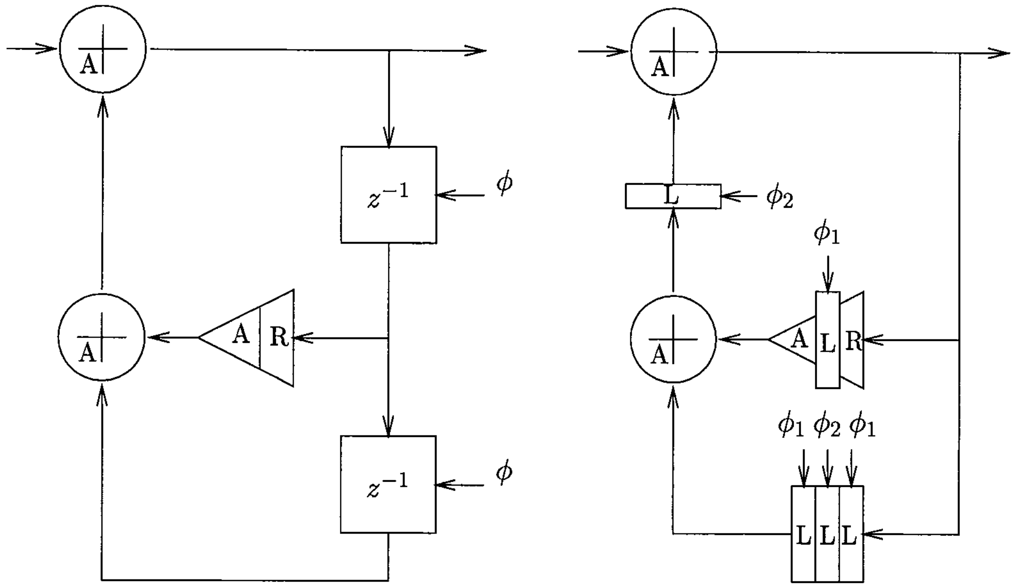


Figure 12: 1.2μ Adder

A two-phase clock permits us to use dynamic latches which are simpler to design and are more compact as compared to 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. 13 (a) and (b). In Fig. 13 (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. 13 (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. 13 (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.

4.5 Other Submodules

Various other macrocells which are needed in our implementation are described here. Half-latch, delay, multiplexer, and inverter are some of the modules which have been designed.



(a) Single phase clock

(b) Two-phase clocking scheme

Figure 13: Clock Speedup

4.5.1 Half-latch with Reset

The schematic for the half-latch is shown in Fig. 14, and the VLSI layout in Fig. 15. It is 16 bits wide corresponding to the internal bus precision. Depending on its location in signal flow graph (see Fig. 17), it latches on either ϕ_1 or ϕ_2. A reset control is also provided. To design the latch, a single bit is laid out, which is tested with Irsim and Spice for functionality and timing. The size of the output invertor is made sufficiently large to allow adequate driving of expected output load in the module where it will be used. Once the testing is completed, the final module is assembled by arraying the one bit. Local distribution of all control signals are taken into account at the 1-bit design stage itself. So after the arraying, the module is ready with all clock/control signals and power rails already wired. Of course, the module is not fully ready until functional verification is done again for the 16-bit wide latch.

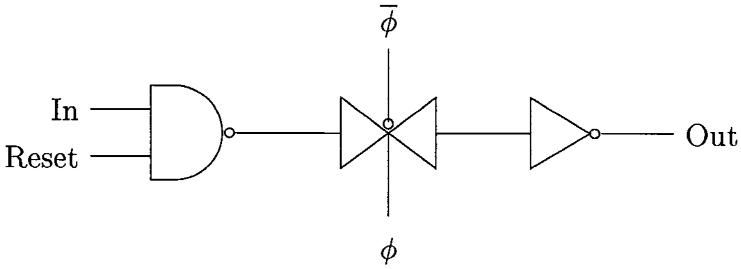


Figure 14: Schematic of Half-latch with Reset Control Circuitry

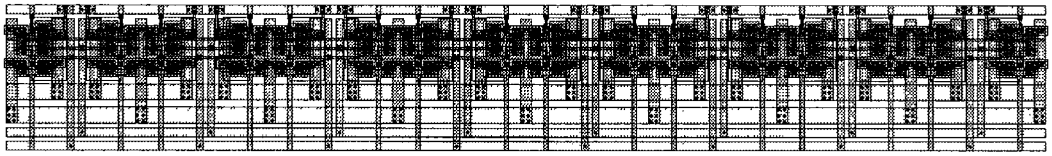


Figure 15: Physical layout of Half-latch with Reset. Module dimension is 938λ × 127λ.

4.5.2 Delay, Delay7 and Delay8

These are shift-registers which act as clock delays for 1, 7, and 8 cycles. As before they are implemented by modifying the basic 1-bit unit to latch for an entire clock period. The 1-bit 1-clock period delay unit is arrayed 16 times to obtain a 16-bit wide bus. And this is further arrayed 7 or 8 times to get the final delay modules. The control/clock signals required by this module are 'Reset', ϕ_1, and ϕ_2. The 'delay' module measures 938λ × 217λ, the 'delay7' measures 1028λ × 1549λ, and the 'delay8', 1028λ × 1771λ.

4.5.3 Multiplexers

These are needed to choose between two sets of signals. It is used after the ROM (see Fig. 17) and other locations in the SFG affected by a switch from computing the forward to the inverse transform. The circuit schematic is illustrated in Fig. 16. The size of the multiplexer is 1271λ × 119λ.

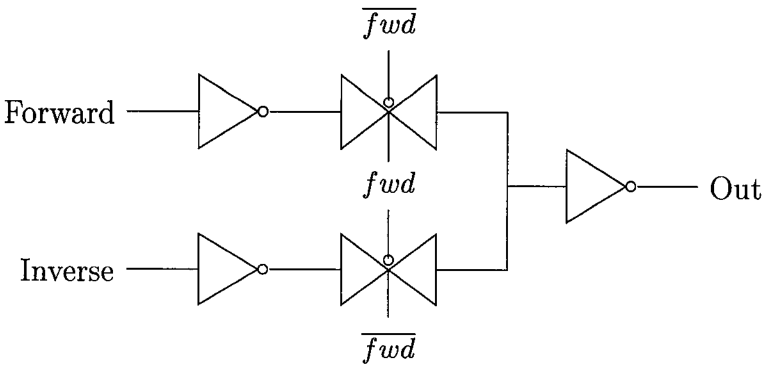


Figure 16: Schematic of Multiplexer

4.6 1-D Channel Module

The macrocells which have been described so far are put together to form the 1-D channel module at the next higher hierarchy. The channel module implements the SFG of the kernel as is shown in Fig. 17. This module, depending on the value stored in the ROM, computes the appropriate DCT/IDCT transform coefficient. Magic 'instantiates' [16] every occurrence of these modules, resulting in much faster layout and extraction of the module.

The signal flow graph implemented by this module is shown in Fig. 17. All but multiplier M_3 (needed only for inverse transform as indicated in Fig. 1) and the associated multiplexers and latches are included in this module. Since only one set of M_3 and its associated circuitry is needed for the entire 1-D module, it is designed separately.

The physical layout of the 1-D channel is shown in Fig. 18. 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 these signals to the entire 1-D module, either from the top or the bottom, as local distribution of these signals is

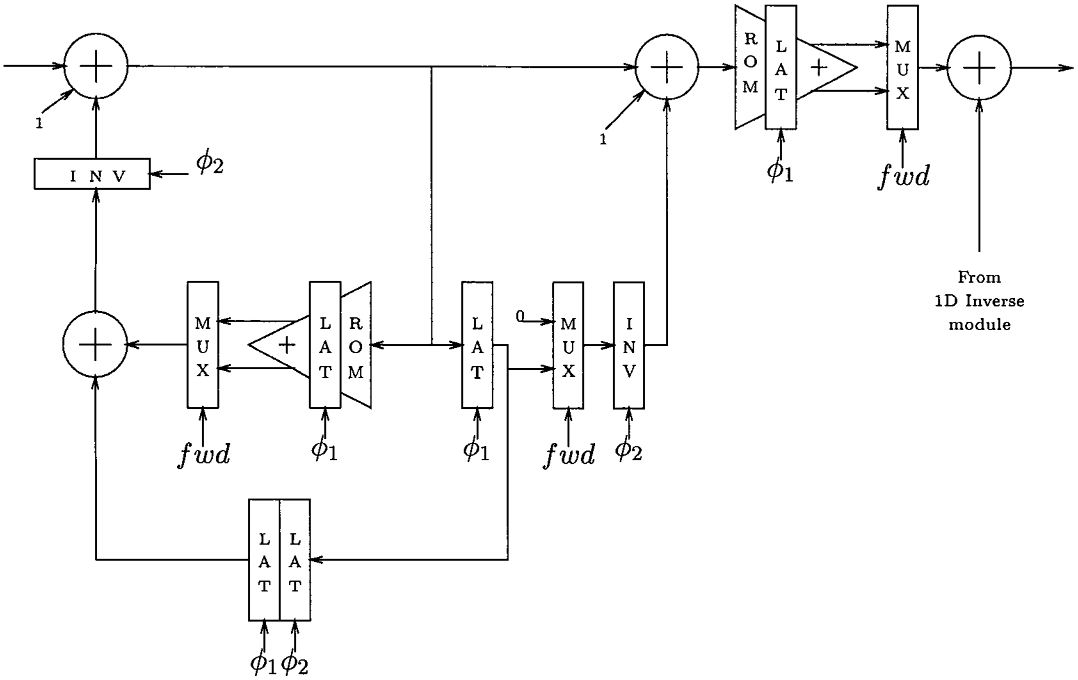


Figure 17: 1-D IIR SFG with clocks and latches

already taken care of in the design of the channel-module.

The eight 1-D channel modules are tiled one over the other to form the complete 1-D DCT. To compute the IDCT, the additional M_3 and its associated delay units is separately attached to the 1-D module, below Channel 7. The physical layout of this module is shown in Fig. 19.

4.7 2-D Channel Module

This module is designed along similar lines as the previous 1-D channel-module. The SFG is in Fig. 20) is almost the same, except that 8-block delay units are present in both loops. This facilitates computation of 8 blocks of data in a time-displaced parallel fashion. The same concept can also viewed from systolic point of view. The VLSI layout of this module is shown in Fig. 21.

4.8 CSA and Control Signal Distribution

This was the last module 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 Circular Shift Array (CSA) module serves another

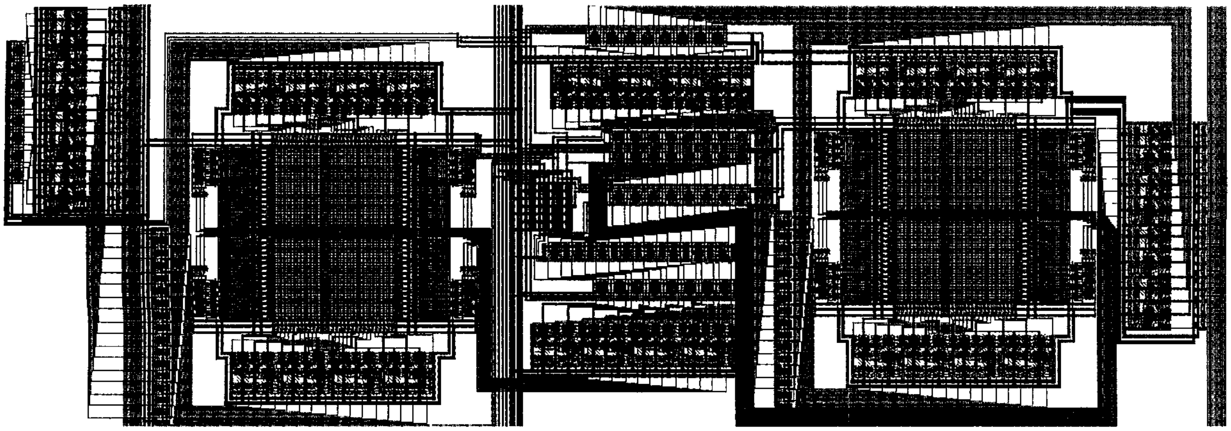


Figure 18: VLSI layout of 1-D channel. The module measures 2742λ × 8029λ.

important function—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. These control signals are Lat_shft_C and Lat_inv_C as given in Table 3. It is during the time the 1-D module is being reset that the 1-D channel outputs are latched into the CSA. At the same time the 2-D DCT is also being computed. In the clock period when the 1-D channel modules are being reset, the 2-D channel modules are not clocked. This is done to ensure synchronicity. For the same reason, the CSA's second set of shift registers which circulate the first row of 1-D DCT coefficients, is also not clocked in that period.

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 clocked modules like latches and delays, and multiplexers. The basic idea is to distribute the master control signals to the high-level cells like 1-D/2-D channel

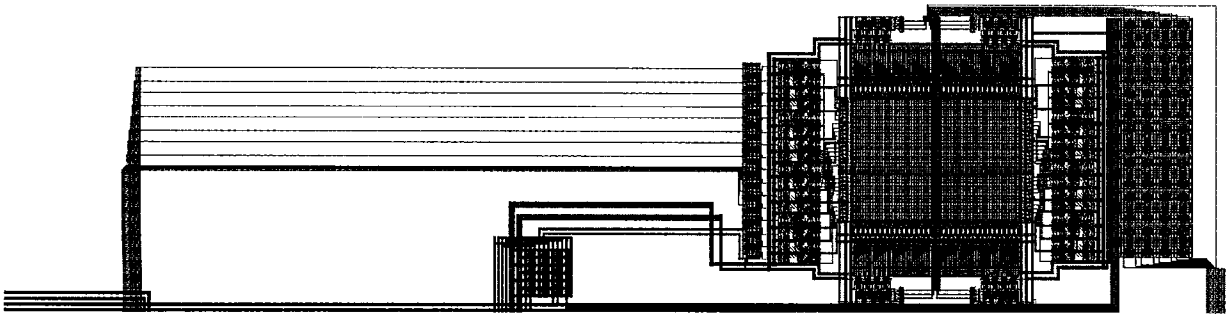


Figure 19: The layout of the additional M_3 module for the inverse transform

modules and use a buffer to generate the local control signals, which are then distributed to all modules within that particular submodule. 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. Those 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. Routing of these control signals to each bit slice is built into the sub-module design.

4.9 2-D DCT/IDCT Chip

Using all the cells—1-D, 2-D, and CSA—already described, the final chip is assembled. The

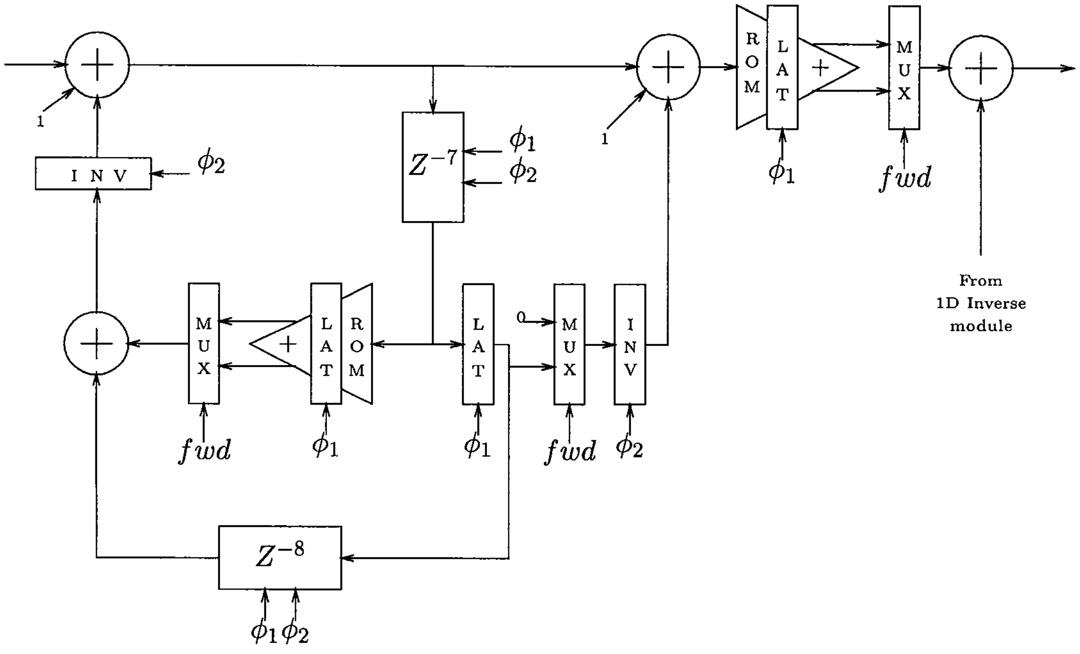


Figure 20: 2-D IIR SFG with latches and clock details

floorplan of the chip is shown in Fig. 22. In the floorplan one can identify the various cells that have been mentioned earlier in this description. The physical layout of the 2-D DCT/IDCT chip is shown in Fig. 23. The chip description and statistics are tabulated in Table 3 and Table 4.

5 Comparisons

Some designs published in the literatures are compared to our design. As can be seen in Table 5, The designs in [5, 8] are for low-bit rate CODECs that operate at about 15 MHz. Both designs used the butterfly architecture so that there is no flexibility in transform size N. The transposition is required. Both employed distributed arithmetic implementation. The DCT/IDCT chip in [6] used of silicon compiler to design the whole system under the 0.8μ triple metal technology. The booth multiplier is used instead of distributed arithmetic. The transposition is also employed. This chip can operate at 50 MHz, but the bit throughput rate is unknown.

It seems that our design has many pins, this is due to ease of debugging. In fact, the active pins are only 38. In particular, a FIFO buffer can be placed at output to reduce output pin count. The chip is a regular, highly modular, and fully-pipelined structure - hence full-custom possible by

Function	Compute 2-D DCT/IDCT
Mode Selection	Active High 'Fwd' for Forward
Clock Pins	Phi 1: ϕ_1 & Phi 2: ϕ_2
Reset Pin (1D)	Active high 'Rst' pin
Reset Pin (2D)	Active high 'Rst_2d_C' pin
CSA Control	'Lat_shft_C' and 'Lat_inv_C'
2-D Inverse	'Inv_load_C' high for last block
Input Pins	ChipIn15, ..., ChipIn0 (16 pins)
Test Output	CSA: 1DOUT15, ..., 1DOUT0 (16 pins)
2-D Output	K0OUT16 — K0OUT1, ..., K7OUT16 — K7OUT1
Power Rails	'Vdd!' & 'GND!'

Table 3: 2D-DCT/IDCT Chip Description

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

Table 4: 2D-DCT/IDCT Chip Statistics

	Sun-Chen [5]	Fujiwara et al [8]	Miyazaki et al [6]	Srinivasan-Liu
Function	16 × 16 DCT	8 × 8 DCT/IDCT	8 × 8 DCT/IDCT	8 × 8 DCT/IDCT
Technology	2.0μ	1.0μ	0.8μ triple metal	1.2μ double-metal
Size	8.3 × 8.1 mm^2	10.7 × 10.2 mm^2	12.8 × 12.6 mm^2	14.7 × 16.2 mm^2
Transistors	73,000	156,000	180,000	320,000
Pads	25 active pads	68 PLCC pkg.	72 PGA	176 (Active pads $\sim$38)
Max. Speed	15.1 MHz	15 MHz	50 MHz	50* MHz or 400* Mbps
Structure	Butterfly (irreg)	Irregular	Irregular	Regular (modular)

Table 5: Comparisons (* The ROM simulation results were based on the 2 μm CMOS technology. Therefore, the actual figures should be higher).

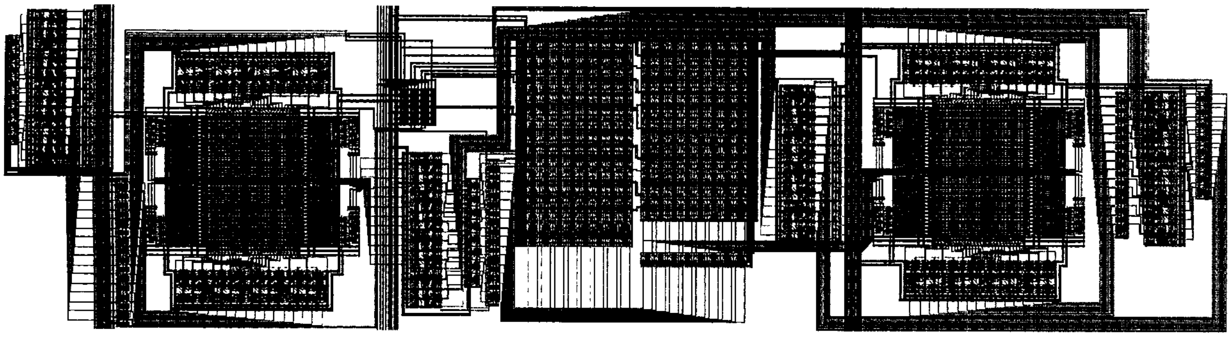


Figure 21: 2-D Module Layout ($2784\lambda \times 10672\lambda$)

using Magic. Based on a conservative estimation it can operate at 50 MHz under the 1.2μ CMOS technology. If 0.8μ technology used, then it can be much faster.

A detailed comparison is given in Table 5.

6 Conclusion

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 2-D DCT/IDCT 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.

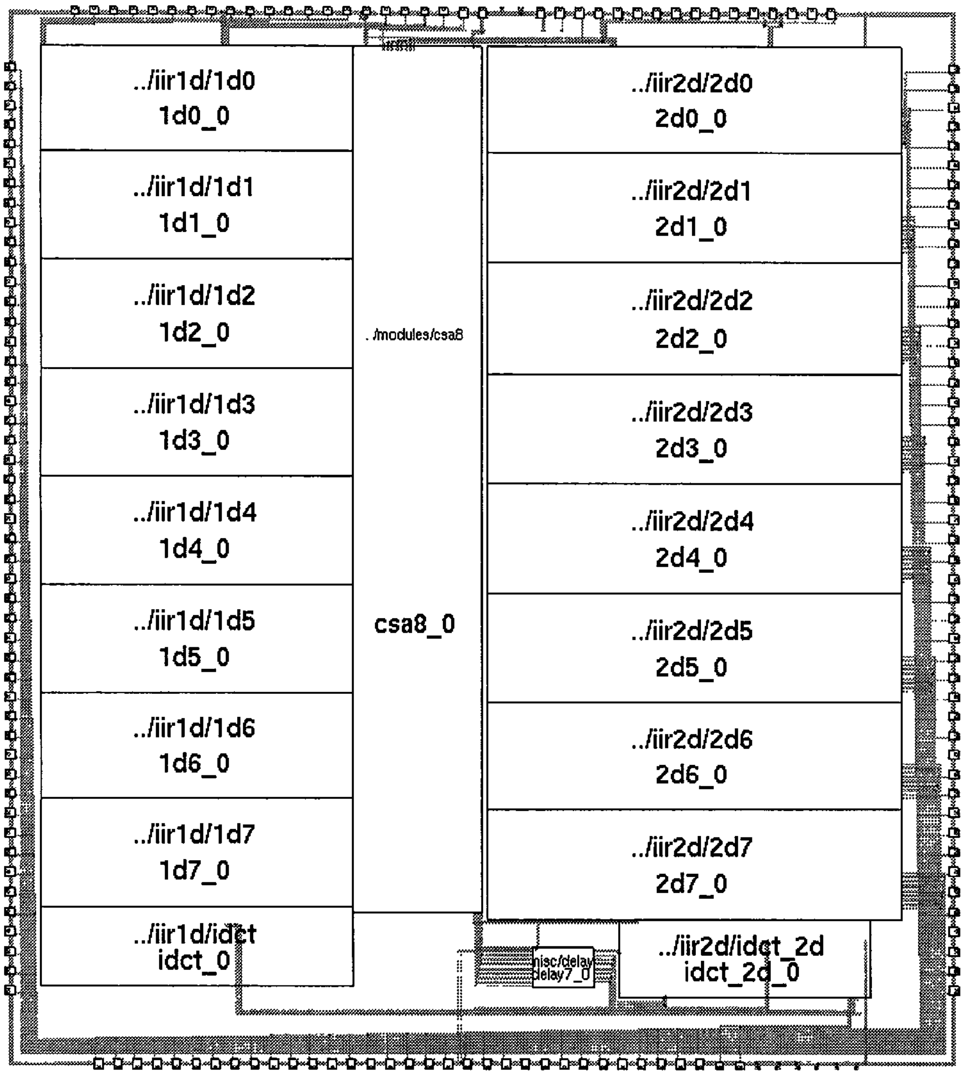


Figure 22: Floorplan of 2-D DCT/IDCT Chip.

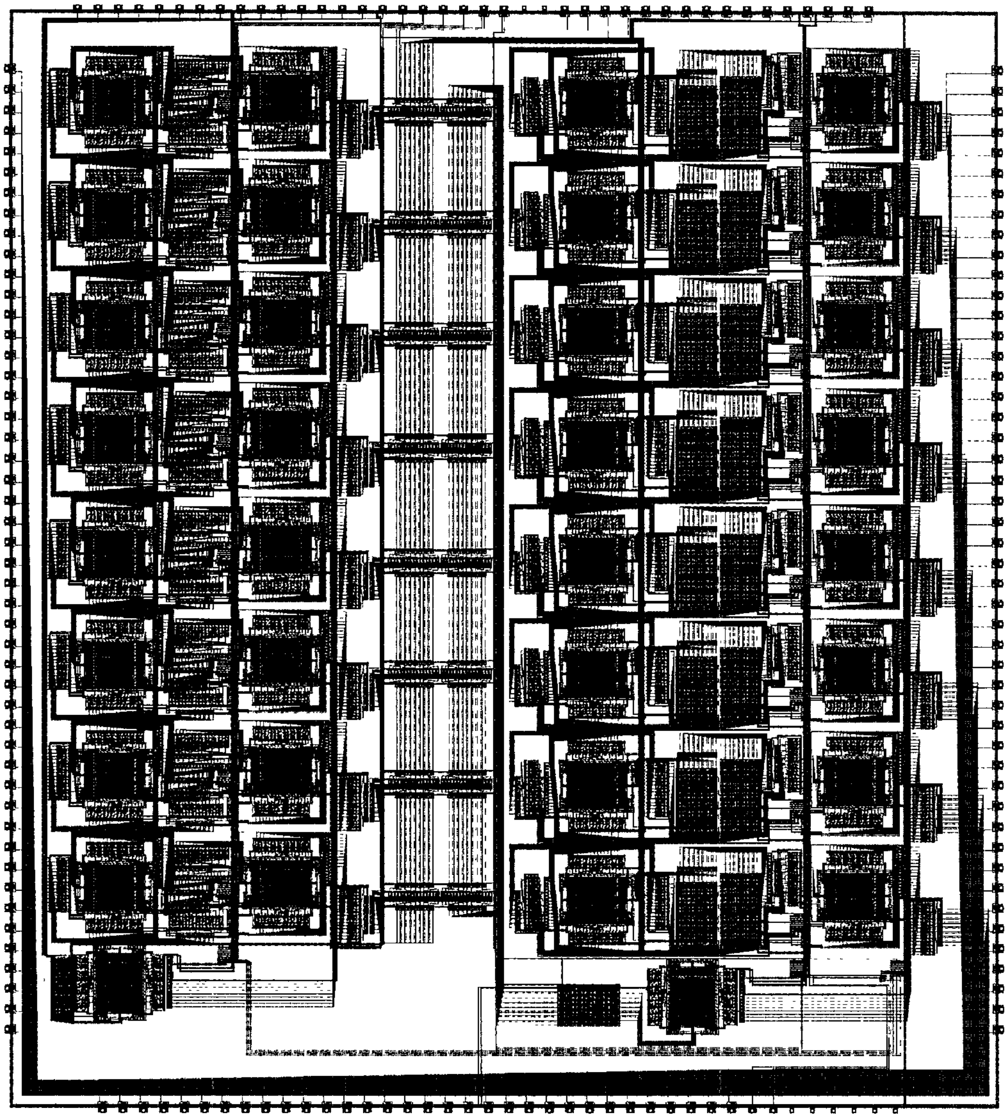


Figure 23: Two Dimensional DCT/IDCT Chip. Chip measures $24550\lambda \times 27094\lambda$. Area is 240 mm^2.

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 with high-speed requirements. In comparison to the existing designs, our approach offers many advantages that can be further explored for even higher performance. This chip has been submitted for fabrication in 1.2μ CMOS N-well double-metal single-poly technology.

References

[1] K. R. Rao and P. Yip, *Discrete Cosine Transform : Algorithms, Advantages, and Applications.* Academic Press, Inc., 1990.

[2] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 31–44, April 1991.

[3] F. Kretz and D. Nasse, "Digital television: Transmission and coding," *Proceedings of the IEEE*, vol. 73, pp. 575–591, April 1985.

[4] G. Tonge, "Image processing for higher definition television," *IEEE Trans. Circuits and Systems*, vol. CAS-34, pp. 1385–1398, November 1987.

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

[6] 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.

[7] M. Vetterli and A. Ligtenberg, "A discrete fourier-cosine transform chip," *IEEE Selected Areas in Communications*, vol. SAC-4, pp. 49–61, January 1986.

[8] H. Fujiwara, M. Liou, and M. Sun, "An all-ASIC implementation of a low bit-rate video codec," *IEEE Trans. Circuits and Systems for Video Techn.*, vol. 2, pp. 123–134, June 1992.

[9] D. Slawecki and W. Li, "DCT/IDCT processor design for high-data rate image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 135–146, June 1992.

[10] 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.

[11] 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.

[12] K. J. R. Liu, C. T. Chiu, R. K. Kologotla, and J. F. JaJa, "Optimal Unified Architectures for the Real-Time Computation of Time-Recursive Discrete Sinusoidal Transforms," IEEE Trans. on Circuits and Systems for Video Technology, vol. 4, pp. 168–180, April 1994.

[13] S. A. White, "Application of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *IEEE ASSSP Magazine*, pp. 4–19, July 1989.

[14] G. Ma and F. J. Taylor, "Multiplier policies for digital signal proessing," *IEEE ASSP Magazine*, pp. 6–20, January 1990.

[15] C. Stearns and P. Ang, "Yet another mulitplier architecture," *IEEE Custom Integrated Circuits Conference*, no. 24, pp. 6.1–6.4, 1990.

[16] R. N. Mayo, M. H. Arnold, W. S. Scott, D. Stark, and G. T. Hamachi, *DECWRL/Livermore Magic Release.* DEC and WRL, Research Report 90/7, 1990.

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

[18] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.