A VLSI Implementation of DCT Using Pass Transistor Technology

S. Kamath, Douglas Lynn and Sterling Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract –**A VLSI design for performing the Discrete Cosine Transform (DCT) operation on image blocks of size 16×16 in a real time fashion operating at 34 MHz (worst case) is presented. The process used was Hewlett-Packard's CMOS26—A 3 metal CMOS process with a minimum feature size of $0.75\mu m$. The design is based on Multiply-Accumulate (MAC) cells which make use of a modified Booth recoding algorithm for performing multiplication. The design of these cells is straight forward and the layouts are regular with no complex routing. Two versions of these MAC cells were designed and their layouts completed. Both versions were simulated using SPICE to estimate their performance. One version is slightly faster at the cost of larger silicon area and higher power consumption. An improvement in speed of almost 20% is achieved after several iterations of simulation and re-sizing.**

1 Introduction

Modern image processing techniques find applications in television broadcasting as well as transmission of pictures from space based artificial satellites. Since a single image requires a tremendous amount of storage or communication channel bandwidth for it's transmission, data compression is employed to reduce this requirement. The Discrete Cosine Transform (DCT) [1] is an important step in many data compression algorithms. The DCT is a frequency domain transformation of a time domain signal which performs very close to the statistically optimal transform—the Karhunen-Loeve Transform (KLT).

In the digital VLSI domain, Complementary Metal Oxide Semiconductor (CMOS) technology is well renowned for it's low power consumption when compared to other logic families. Traditional CMOS technology uses both PMOS and NMOS transistors for forming logic and hence the name 'Complementary'. In CMOS pass technology [7, 8, 9, 12, 13] however, NMOS transistors alone are used for forming the logic. Both PMOS and NMOS transistors are employed only in the inverters which serve the dual purpose of inverting logic as well as buffering signals. The inverter buffers are needed to restore the quality of the signal passed by an NMOS transistor since it passes a degraded '1'. The most important reason for using pass transistor technology is the area savings that result when compared to the traditional CMOS [8, 9]. The pass transistor based VLSI layouts are dense and compact and in general most of the chip area tends to be the area devoted to forming the logic itself.

This paper presents the design and VLSI implementation of a major portion of a two dimensional DCT chip proposed to NASA to be incorporated in one of their space based image compression applications. Since the emphasis is on speed, we have tried to improve the speed of the chip while maintaining power and area requirements within acceptable limits. The design steps are reviewed in the following sections on a block diagram level. The transistor level circuit diagrams are available in [4]. Some of the important layout aspects are mentioned with details and actual layouts being available in [4]. A section on design verification is included with explanations on how an improvement of nearly 20% in speed is achieved without any consequent increase in area. To conclude, the salient features of the chip designed are summarized in a table along with notes on what remains to be done to complete the chip before it can be fabricated.

2 Circuit Design

The design of 2D DCT chip is explained on a block diagram level in the following three subsections. The transistor level circuit diagrams used in the design are available in [4].

2.1 Definition of DCT and overall chip description

The Discrete Cosine Transform (DCT) [1] of a finite data sequence which in this case is the real time image data $a[j]$ of length N is defined as

$$C(u) = \frac{2\ m(u)}{N} \sum_{j=0}^{N-1} a[j]\ cos\left\{\frac{(2j+1)u\pi}{2N}\right\} \quad where\ u = 0, 1, \ldots, (N-1) \tag{1}$$

$$and \quad m(u) = \begin{cases} \frac{1}{\sqrt{2}} & if\ u = 0 \\ 1 & if\ 1 \le u \le (N-1) \end{cases}$$

The above definition applies to one dimensional (1D) data sequence.

For a two dimensional data sequence, like the two dimensional (2D) image data $a[j,k]$ having the dimensions of $N \times N$, a 2D DCT is defined as

$$C(u,v) = \frac{4\ m(u)\ m(v)}{N^2} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} a[j,k]\ cos\left\{\frac{(2j+1)u\pi}{2N}\right\}\ cos\left\{\frac{(2k+1)v\pi}{2N}\right\} \tag{2}$$

$$where\ u = 0, 1, \ldots, (N-1)\ \ \&\ \ v = 0, 1, \ldots, (N-1)$$

Here the values of the scale factors $m(u)$ and $m(v)$ are defined as was $m(u)$ in equation (1).

Assuming the image is divided into blocks of 16×16 pixels, $a[j,k]$ in equation (2) becomes a 16×16 matrix. Equation (2) can be partitioned so that the 2D DCT is performed essentially as two 1D DCTs, the first operating on the rows of the image data and the second on the transpose of the matrix resulting from the 1D DCT operations. Assuming that the pixel data is input to the chip row-wise one pixel (word) at a time, the proposed method performs a 1D DCT on each row of the input data. This takes 16 clock cycles per row assuming data is presented at the rate of one word per clock cycle and assuming that a multiply-accumulate operation can be performed in one clock cycle. The multiplication referred to here is the multiplication of the pixel data with the *cos* terms in equation (2) appropriately scaled.

Since the scaled *cos* terms can be pre-computed they are treated as constant coefficients and stored in memory. For the DCT to be computed in real time, assuming the pixel data is a continuous never-ending stream of data, we need 16 multiplier-accumulators (MACs). Each MAC is responsible for computing one of the 16 transform domain terms using equation (1). After 16 × 16 cycles, all the row DCTs will be completed. Then the DCTs on the columns can begin. Assuming again that real time operation is required, a second set of 16 MACs is needed. When the column DCTs are being computed, row DCTs can be referred to the second 16 × 16 image block. This requires a 16 × 16 buffer memory (called the transposition memory) between the two sets of MACs. Each set of MACs can be configured as either a 4×4 array or an 8×2 array depending on the size of the transposition memory. The overall block diagram of the chip is shown in Figure 1.

The ROM cells alongside each of the MACs are arrays of Read Only Memories with each cell containing an unique column of the coefficient matrix (scaled *cos* terms from Equation 2). The transposition memory performs a transpose operation and provides the transposed 1D DCT output to the second dimension processing MACs one word at a time as shown in Figure 1. Image data enters the chip from the top left one word at a time and the 2D DCT output from the chip leaves the chip one word at a time through the shift register at the bottom right. The shift register is a parallel input (word) serial output (word) register.

The multiply operation for the DCT is accomplished in the MAC cell by using modified Booth recoding. The function of the MAC can be expressed mathematically by the following recursive equation.

$$c[i] = a[i] \cdot b[i] + c[i-1] \quad for\ i = 0, 1, \ldots, 15 \tag{3}$$

Here the $a[i]$ input is the image data and $b[i]$ input is the scaled cos term in equation (1). At the beginning of the process when $i = 0$, $c[i-1]$ is set to 0 with $a[0]$ and $b[0]$ being the first element of the first row and the first column of the image matrix and the coefficient matrix respectively. For the next 15 clock cycles when $i = 1, \ldots, 15$, the c output from the previous multiply-accumulate operation becomes the present c input. Thus, each MAC accumulates the inner product of one row of image data and one column of the coefficient matrix.

This method was chosen mainly due to it's speed and regularity of implementation. The VLSI layouts for two versions of the MAC were completed and both were found to be highly area efficient and neither version required any complex routing. In the proposed design due to accuracy considerations, a 16 × 16 bit multiplier with accumulate capacity was chosen. Both versions of the multiplier cells are considered in detail in the next subsection.

2.2 Design of Modified Booth Recoded MAC

The process of parallel multiplication of two multi-bit numbers, say A and B, consists of multiplying all the bits of A by each of the bits of B starting from the Least Significant Bit (LSB) and shifting the resulting partial products one bit to the left for every operation before performing the final addition of the properly shifted partial products to form the final answer. The modified Booth algorithm reduces the number of partial products that must be added together, thereby speeding up the computation [2] by recoding the bits of B. The exact magnitude of the speed up is a function of the number of bits the algorithm considers

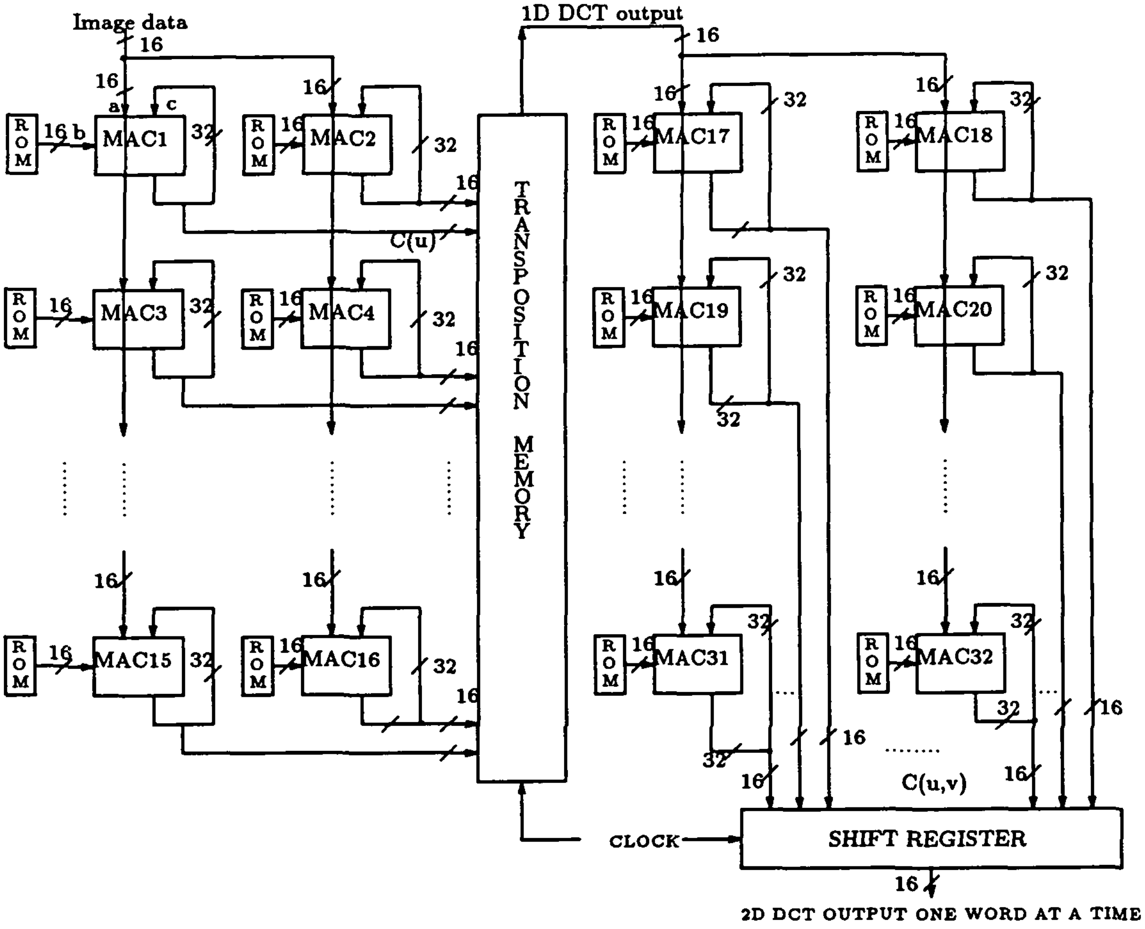

Figure 1: Overall block diagram of the 2D DCT chip

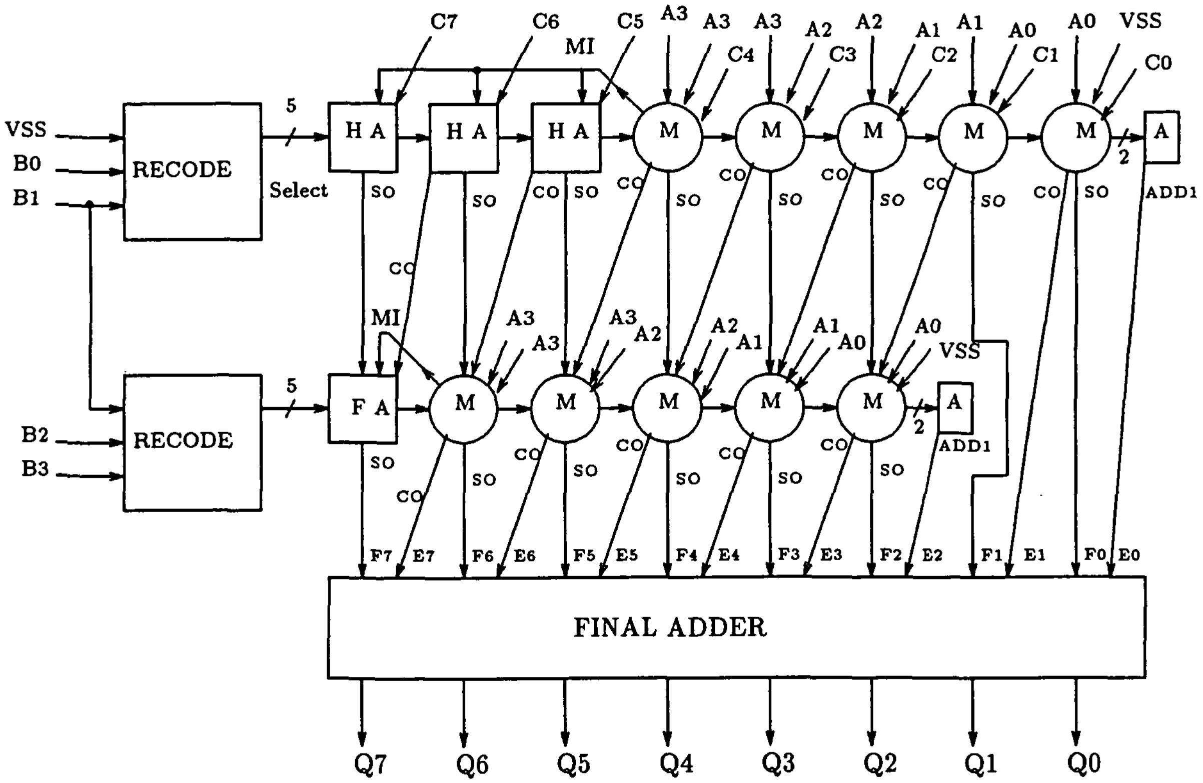

Figure 2: Block diagram of a 4 × 4 bit Booth MAC

in each step. In this design, a bit-pair recoding is employed as a result of which the number of rows of partial products to be added together is cut in half thereby yielding a speed up factor of nearly 50%. The recoded digits take on values ranging from -2 to +2 depending on the combination of three bits of B including one overlap bit for every step. The details about this recoding scheme are available in [2] and design steps for it's implementation in pass transistor technology are summarized in [4].

The block diagram of 4 × 4 bit Booth MAC is shown in Figure 2. Since the recoding algorithm requires that the partial products be shifted 2 positions to the left for every row of partial product, the array takes a trapezoidal shape. The circular blocks labeled M are the multiplier cells, the HA blocks are simple Half-Adders, the FA block shown is a Full-Adder, the blocks denoted as A are the Add 1 cells and finally the RECODE blocks are the recoding circuits for the B operand.

The multiplier cell is responsible for selecting the appropriate partial product bit and adding this to two other bits namely, the Sum-In (SI) and the Carry-In (CI) as shown in Figure 3. The SI and the CI are the Sum-Out (SO) and Carry-Out (CO) signals from appropriate cells of the row above the multiplier cell in question. The partial product bits presented to each multiplier cell are A_i (A) and A_{i-1} (AM1) with their complements being provided by inverters within the multiplier cell. The outputs of the multiplier cell namely, the SO and the CO signals become the SI and CI inputs to the appropriate cells in the row below it. This carry save technique [11], is depicted in Figure 2. The multiplier cells in the top row contain Half-Adders where as the multiplier cells in all the other rows use

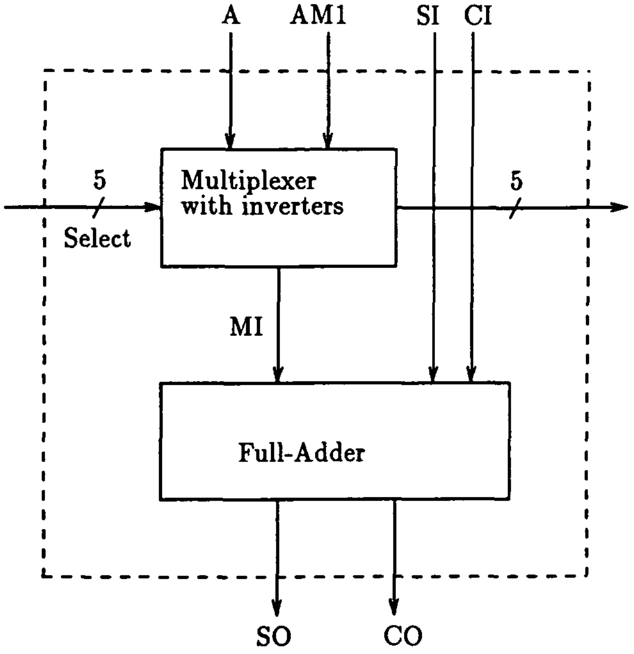

Figure 3: Block diagram of a multiplier cell

Full-Adders. The signal labeled MI takes care of the sign extension.

The design steps for implementing the above cells in pass transistor technology are available in [4]. In pass transistor implementation we need both the signal and it's complement. There are two ways in which this can be done within the MAC array. One option is to simultaneously generate the complements of signals using redundancy in hardware. This option (option A), increases the individual cell areas in the MAC, thereby increasing the area of the MAC cell itself. Option B reduces hardware redundancy thereby saving some of the area of a multiplier cell by using additional inverters to obtain complements of signals. Figure 4 illustrates the multiplier cells designed using both options. The area of these cells and their comparative speed etc. are given in subsequent sections. In this figure the block named BTS is a Binary Tree Structured [6] network and INV(1) as well as INV(2) represent a single and double inverters respectively. The block labeled MUX is a 5:1 multiplexer.

2.3 Conditional Sum adder for the final addition

An interesting scheme called the conditional sum adder scheme [10] has been investigated by K. Cameron and J. Canaris [3] and changes in the design to suit CMOS pass transistor version have been proposed by them. This type of adder is found to be extremely fast particularly for large word sizes. This adder has since been incorporated in a VLSI chip fabricated using a 1.6μm double metal CMOS process. The same approach has been adopted in this design with appropriate changes in transistor sizing and VLSI layouts to suit the 1μm drawn (0.75μm internal) three metal CMOS process that has been used here. As expected, this not only saved chip area but after re-sizing some of the transistors reduced the worst case delay through the adder as compared to the earlier design. These details are included in the subsequent sections.

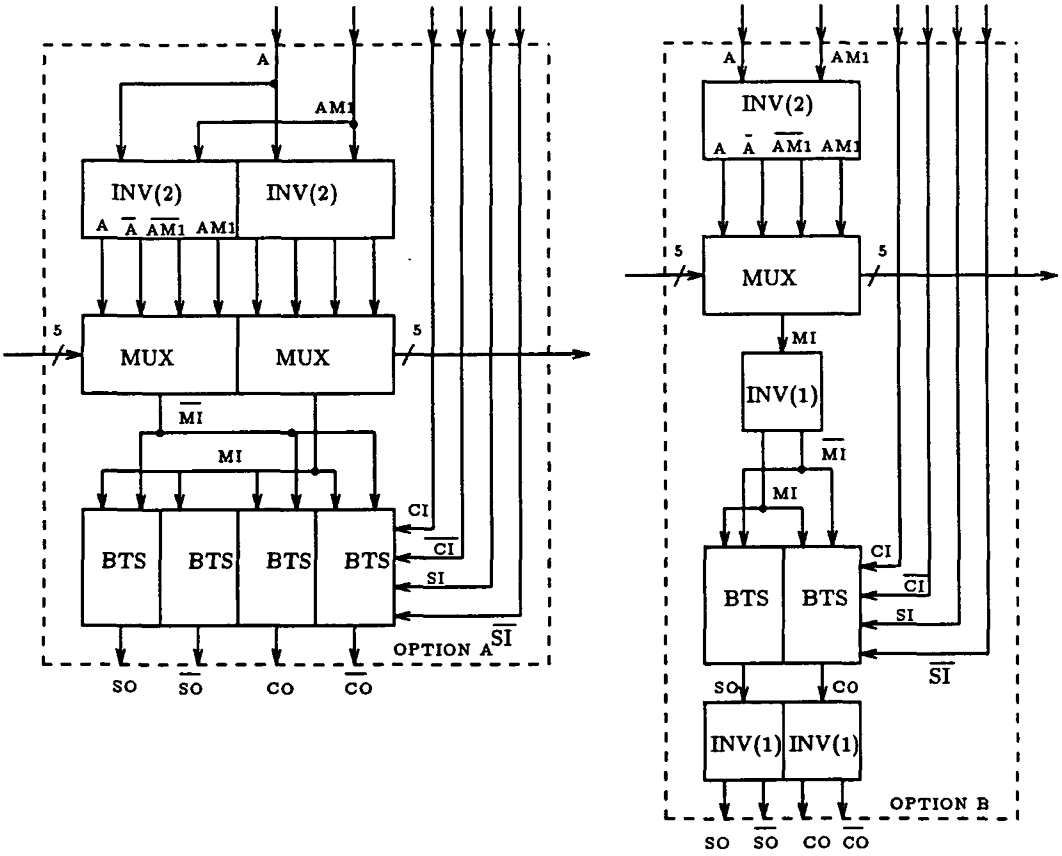

Figure 4: Block diagrams of the multiplier cell for option A and option B

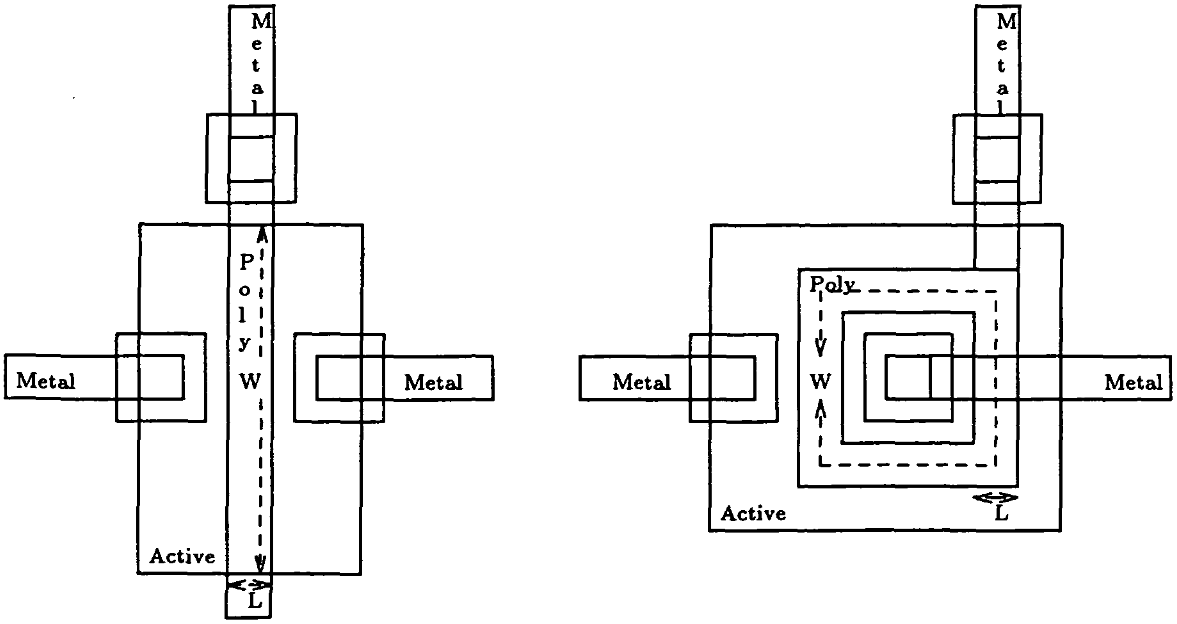

Figure 5: Typical layouts of a regular and square transistor

3 Layouts

The following two subsections deal briefly with the layouts of some of the cells required for the multiplier section and the conditional sum adder section.

3.1 Layouts of multiplier cells

After taking into account the various VLSI issues discussed in detail in [4], the layouts for multiplier cells for option A and option B were drawn. The areas of the cells in option A and option B are $69\mu m \times 69\mu m$ and $42.2\mu m \times 76.6\mu m$ respectively which translates to an area saving of 32% for option B when compared to option A.

3.2 Layouts of conditional sum adder

As mentioned earlier, the conditional sum adder was previously designed using a $1.6\mu m$ double metal process [3]. Since the process used for this design had a minimum feature size of $1\mu m$ (drawn) which after the shrinkage was reduced to $0.75\mu m$, effectively there was a reduction in the minimum lengths of transistors by a factor slightly more than two. This implied that the widths of transistors could be reduced by the same factor without any appreciable change in the performance of the transistor. Doing this resulted in the adder taking about 25% less area when compared to the earlier design.

Square transistors were used in certain inverter buffer cells to take advantage of the resulting reduction in capacitance thereby improving the speed. Figure 5 contrasts the regular and square transistors. More details about these two ways of drawing a transistor along with their merits and demerits are available in [4].

$\overline{SO}$ signal	Delay for option A (ns)	Delay for option B (ns)
First row	4.27	4.5
Second row	5.88	6.5
Third row	7.65	8.5
Fourth row	9.41	10.4
Fifth row	11.32	12.8

Table 1: The worst case delay through the multiplier section

The approximate areas of the multiplier section in option A and option B are $2300\mu m \times 800\mu m$ and $1600\mu m \times 850\mu m$ respectively. The area of the conditional sum adder is approximately $1150\mu m \times 210\mu m$. Thus, when the multiplier and conditional sum adder sections are combined, option B is not only more area efficient, but also less area is wasted as compared to option A.

4 HPSPICE simulation

Representative portions of the multiplier section for both options and the 32 bit conditional sum adder are simulated using HPSPICE to estimate their worst case propagation delay as well as the approximate dynamic power dissipation in them. The worst case conditions for this design are assumed to be 4.3V VDD and 0.3V VSS after accounting for the noise on the power supply and a temperature of 110C.

4.1 Simulations on the multiplier section

Representative portions of the multiplier sections for both options were simulated for several worst case input patterns. The worst case timing patterns were determined by [5] for a similar structure. Since the sum path through the multiplier array turned out to be the critical delay path, the input pattern which resulted in only the sum signals transitioning as a result of input signals transitioning was found to be the worst among the worst case patterns analyzed. This pattern was $A = 1111\ 1111\ 1111\ 1111$ and $B = 1000\ 0000\ 0000\ 0001$ and $C = 0$. None of the carry signals transitioned during this simulation. Table 1 summarizes the delays for $\overline{SO}$ signals for the top five rows of the multiplier section for both the options. This table shows that option A is slightly faster than option B by about 1 ns for four rows which can be extrapolated to 2 ns for the entire multiplier section.

An approximate estimate for the average dynamic power dissipated in the multiplier section of the 32 bit MAC is obtained as 47.4 mW for option A and 29.8 mW for option B at the maximum operating speed.

4.2 Simulation of conditional sum adder

The conditional sum adder was earlier designed using pass transistor technology using a $1.6\mu m$ double metal CMOS process with a worst case propagation delay of 24 ns [3]. In this design the corresponding number after the first-cut simulation using standard transistor

Signal name	conditional sum adder delay (ns)	MAC delay (ns)
Q3	6.83	10.96
Q7	9.85	15.56
Q15	13.82	24.19
Q31	18.9	29.39

Table 2: The worst case delays through the conditional sum adder and the MAC

layouts everywhere was 23 ns assuming a load capacitance of 1 pF. Using square transistors in two of the inverter buffer cells namely, MUXBUFF2(D) and MUXBUFF2(E), the worst case delay was reduced to 22 ns. Further analysis of the simulation results and after resizing the transistors in MUXBUFF2(E) to reduce the capacitive load on the critical delay path, the delay was reduced to 18.9 ns which translated into a speed improvement of nearly 20%.

In order to estimate the performance of the MAC, information from two separate simulations namely, the multiplier section alone and the 32 bit conditional sum adder alone was used. Since the worst case output timing pattern for each output of the multiplier section is predictable as per Table 1, the timing of the inputs to the conditional sum adder were all shifted in time by the same amounts as they would be if they were to be the outputs of the multiplier section. The simulation that followed yielded a worst case delay through the conditional sum adder and due to the time shifting of inputs, the delay through the entire MAC itself as 29.39 ns. This translated into a clock frequency of 34 MHz for the DCT chip. Table 2 summarizes the delay for the various bit positions of the conditional sum adder outputs alone and with it's inputs shifted in time to mimic the performance of the MAC.

An estimate for the average dynamic power consumption in the conditional sum adder was 10.25 mW. Thus the approximate figure for the MAC turned out be 57.7 mW for option A and 40 mW for option B.

5 Conclusions

Critical blocks for a pass transistor based CMOS 2D DCT chip were designed and performance estimated. An improvement of nearly 20% in speed was obtained over previous work. The processing of the image data was performed parallelly using 16 MACs for 1D processing and an equal number of MACs for the second dimensional processing. The two 16 × 16 matrices representing the real-time image data and the coefficients were multiplied using MAC cells the layouts of which were regular and there were no complex routings. The MAC employed the modified Booth recoding algorithm and performed the multiplication in 2's complement arithmetic. A 32 bit conditional sum adder was used inside each MAC for the purpose of final addition. This adder was found to be extremely fast with the minimum number of series transistors in it's critical path thereby making it one of the fastest adders for the word size among various types of adders investigated. This was verified by simulations on the actual layouts using a software tool for VLSI design wherein after the resizing of some of the inverter buffers to improve the speed by nearly 20%, the final adder overhead for one of the worst case delays turned out to be only 10.5 ns. Table 3 summarizes the salient features of the chip. Although none of the blocks in Figure 1 other than the MAC

Technology	Digital CMOS
Process	CMOS26 (3 metal)
Minimum length	$1\mu m$ (drawn) $0.75\mu m$ (internal)
Core area for 32 MAC cells	6.4mm × 6.8mm (option B)
Transistor count for 32 MAC cells	247,904 (option B)
16×16 bit worst case multiply time	29.39 ns (option B)
Worst case clock frequency	34 MHz. (option B)
Ave. dynamic power consumption (32 MACs)	1280 mW (option B)
Number of pins	32 + power, ground, clock etc.
DCT rate	$\frac{34MHz}{16x16}$

Table 3: The salient features of the chip designed

has been exclusively designed and simulated for this design, some estimates are available for the design and implementation of these blocks. The ROM cells placed along side each of the MAC cells need to store 256 bits of data each. From an earlier design in which ROM cells were used [14] it was possible to estimate the area required for each ROM after adjusting for the process as $200\mu m \times 140\mu m$. This was found to be a very small percentage of the area of the MAC cell. Likewise it is envisaged that it would not take too much of silicon area to implement the transposition memory, shift register etc shown in the overall block diagram in Figure 1 and it is expected that it would be easy to pitch match these blocks to the existing MAC layouts. It would then remain to provide buses for the power supplies, input output pads, pad drivers, clock, clock drivers etc. before the chip could be fabricated. Even if all these items meant doubling the core area for 32 MACs as a rough estimate for the entire 2D DCT chip, the chip size would still remain a very practical and realizable entity.

6 Acknowledgements

This research was supported in part by NASA under Space Engineering Research Grant NAGW-1406 and by the NSF under Research Initiation Grant MIP-9109618.

References

[1] N. Ahmed, T. Natarajan and K. Rao, " Discrete Cosine Transform", IEEE Transactions on Computers, Jan. 1974, pp. 90-93.

[2] M. Annaratone, *Digital CMOS Circuit Design*, Kluwer Academic Publishers, Boston, MA, 1986.

[3] J. Canaris and K. Cameron, "A Comparison of Two Fast Binary Adder Configurations", NASA SERC 1990 Symposium on VLSI Design, Moscow, ID, pp. 78-86.

[4] S. Kamath, "A VLSI implementation of Discrete Cosine Transform using Pass Transistor Technology", M.S Thesis, University of Idaho, Moscow ID.

[5] Y. Oowaki et al., "A 7.4ns CMOS 16 × 16 Multiplier", 1987 IEEE International Solid-State Circuits Conference, pp. 52-53.

[6] G. Peterson and G. Maki, "Binary Tree Structured Logic Circuits: Design and Fault Detection", Proceedings of IEEE International Conference on Computer Design: VLSI in Computers, Port Chester, NY, Oct., 1984, pp. 671-676

[7] D. Radhakrishnan and G. Maki, *Design of Pass Transistor Switching Circuits*, Moscow, ID, University of Idaho, 1983.

[8] D. Radhakrishnan, S. Whitaker and G. Maki, "Formal Design Procedures for Pass Transistor Switching Circuits", Proceedings IEEE Custom Integrated Circuits Conference, Rochester, NY, May, 1984, pp. 139-144

[9] D. Radhakrishnan, S. Whitaker and G. Maki, "Formal Design Procedures for Pass Transistor Switching Circuits", IEEE JSSC, vol. SC-20, April, 1985, pp. 531-536

[10] A. Rothermal et al., "Realization of Transmission Gate Conditional Sum (TGCS) Adders with Low Latency Time", IEEE Journal of Solid State Circuits, vol. 24, No. 3, June 1989, pp. 558-561.

[11] S. Steinlechner and G. Spahlinger, "Carry-Save Adders and their Application for a Multiplication with Factored Multiplicands", Proceedings of IEEE International Conference on Computer Design: VLSI in Computers, Port Chester, NY, Oct., 1985, pp. 359-362

[12] S. Whitaker, *Design of Combinational Logic with Pass Transistors*, Moscow, ID, University of Idaho, 1982.

[13] S. Whitaker, "Pass Transistor Networks Optimize NMOS Logic", Electronics, McGraw Hill, New York, NY, Sept. 22, 1983, pp. 144-148

[14] S. Whitaker, J. Canaris and K. Cameron, "Reed Solomon VLSI Codec For Advanced Television", IEEE Transactions on Circuits and Systems for Video Technology, vol. 1, No. 2, June 1991, pp. 230-236.