9-1-1992

VLSI Implementation of Discrete Cosine Transform Based on the Shared-Multiplier Algorithm

Sze-Po Hung
Purdue University School of Electrical Engineering

Wen-Zen Shen
Purdue University School of Electrical Engineering

O. K. Ersoy
Purdue University School of Electrical Engineering

VLSI Implementation of Discrete Cosine Transform Based on the Shared-Multiplier Algorithm

Sze-Po Hung
Wen-Zen Shen
Okan K. Ersoy

TR-EE *92-40*
September *1992*

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285

VLSI Implementation of Discrete Cosine Transform Based on the Shared-Multiplier Algorithm

Sze-Po Hung, Wen-Zen Shen and O.K. Ersoy

Purdue University
School of Electrical Engineering
West Lafayette, IN 47907

Abstract

In this paper a new algorithm for discrete cosine transform (DCT) is proposed. This algorithm is especially efficient for VLSI implementation because each multiplier in the 1-D DCT is shared by two constants rather than one. This greatly reduces the chip area, and the high speed characteristics are still retained. Based on this algorithm, we have developed the corresponding bit-parallel, fully-pipelined architecture for the size-8 DCT. The core area of the chip is only 8.6mm x 8.5mm, using 1.2um double-metal single-poly CMOS technology. This chip is simulated for operation at the maximum speed of 100 MHz which far exceeds the speed requirement of the HDTV system (70 MHz).

I. Introduction

The applications of the digital speech and image signals are more and more extensive today than ever before. They usually go through compression before transmission to reduce the bandwidth. The discovery of the discrete cosine transform (DCT) in 1974 [1] provided a significant impact on DSP field. It is believed to be one of the most powerful data compression tools. Consequently, DCT is extensively used in visual communications.

In the past decade many papers have been published concerning about reducing the computation complexity, especially the number of multiplications, of DCT. Most of them are efficient with software implementation, but unfortunately due to their irregular structure and complex routing, these algorithms are not suitable for VLSI implementation. Our goal is to design a high speed 8x8 DCT chip suitable for the application in HDTV system. Aimed at this, we derived an algorithm suitable for VLSI implementation. This algorithm is referred to as " the shared-multiplier algorithm.

In Section 11, we demonstrate the derivation of our algorithm. In Section 111, several well-known DCT computation algorithms are compared. The flow-graph of 8x1 DCT based on our algorithm is also shown in Section III. In Section IV, we introduce a bit-parallel, fully-pipelined architecture for size-8 DCT. The layout of the final circuit is shown in Section V. Conclusions are given in Section VI.

II. Shared-multiplier algorithm

The one-dimensional DCT can be expressed as

$$Y(k)=c(k)\sum_{n=0}^{N-1} x(n)\cos\frac{(2n+1)k\pi}{2N} \quad (1)$$

where $c(k)=1/\sqrt{2}$ $k=0$

$=1$ otherwise

and N is assumed to be even.

Defining $u(n)=x(n)+x(N-n-1)$ (2a)

$$v(n)=x(n)-x(N-n-1) \quad (2b)$$

where $n=0,1, \ldots ,N/2-1$.

Eqn(1) can then be divided into two parts --- odd and even terms as follows:

$$Y(k)=c(k)\sum_{n=0}^{N/2-1} u(n)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad (3)$$

where $k=0,2, \ldots,N-2$

$$Y(k)=c(k)\sum_{n=0}^{N/2-1} v(n)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad (4)$$

where $k=1,3, \ldots ,N-1$

Look at the following derivation:

$$\cos\left[\frac{(2n+1)k\pi}{2N}\right]+j\cos\left[\frac{(2n+1)(N-k)\pi}{2N}\right]$$
$$=\cos\left[\frac{(2n+1)k\pi}{2N}\right]+j(-1)^n\sin\left[\frac{(2n+1)k\pi}{2N}\right]$$
$$=e^{j(-1)^n(2n+1)k\pi/2N}$$
$$=W_{4N}^{(-1)^n(2n+1)k} \quad (5)$$

where $W_N^k=\exp(j2nk\pi/N)$ (6)

Defining

$$A(k)=\mathrm{Re}\left\{c(k)\sum_{n=0}^{N/2-1}u(n)W_{4N}^{(-1)^n(2n+1)k}\right\} \quad (7)$$

$$B(k)=\mathrm{Im}\left\{c(k)\sum_{n=0}^{N/2-1}u(n)W_{4N}^{(-1)^n(2n+1)k}\right\} \quad (8)$$

for $k=0,2,\ldots,N/2$

Assuming the input sequence is real, then

$$\begin{aligned}A(k)&=c(k)\sum_{n=0}^{N/2-1}u(n)\mathrm{Re}\left\{W_{4N}^{(-1)^n(2n+1)k}\right\}\\&=c(k)\sum_{n=0}^{N/2-1}u(n)\cos\left[\frac{(2n+1)k\pi}{2N}\right]\\&=Y(k)\end{aligned} \quad (9)$$

While

$$\begin{aligned}B(k)&=c(k)\sum_{n=0}^{N/2-1}u(n)\mathrm{Im}\left\{W_{4N}^{(-1)^{2n+1)k}}\right\}\\&=c(k)\sum_{n=0}^{N/2-1}u(n)\cos\left[\frac{(2n+1)(N-k)\pi}{2N}\right]\\&=Y(N-k)\end{aligned} \quad (10)$$

Therefore

$$Y(k)+jY(N-k)=c(k)\sum_{n=0}^{N/2-1}u(n)W_{4N}^{(-1)^n(2n+1)k} \quad (11)$$

for $k=0,2,\ldots,N/2$

Similarly

$$Y(k)+jY(N-k)=c(k)\sum_{n=0}^{N/2-1}v(n)W_{4N}^{(-1)^n(2n+1)k} \quad (12)$$

for k=1,3, ... , N/2-1

Assuming 4 is a factor of N,eqn(11) can be rewritten as

$$Y(k)+jY(N-k)=c(k)\{\sum_{n=0}^{N/4-1} u(2n)W_{4N}^{(4n+1)k} + \sum_{n=0}^{N/4-1} u(2n+1)W_{4N}^{(4n+3)k}\} \quad (13)$$

Letting m=N/4-I-n, we have

$$Y(k)+jY(N-k)=c(k)\{\sum_{n=0}^{N/4-1} u(2n)W_{4N}^{(4n+1)k} + \sum_{m=N/4-10}^{0} u(N/2-1-2m)W_{4N}^{(N-1-4m)k}\} \quad (14)$$

Changing the variable and factoring out W_{4N}^{k}, we have

$$Y(k) + jY(N - k) = c(k)\left[\sum_{n=0}^{N/4-1}\left[u(n) + W_4^{-k}u(N/2-1-2n)\right]W_N^{nk}\right] \quad (15)$$

Introducing a new variable d(n,k)

$$d(n,k)=u(2n)+W_4^{-k}u(N/2-1-2n) =u(2n)+(-1)^{k/2}u(N/2-1-2n) \quad \text{,for even k} \quad (16)$$

Eqn(15) can then be written as

$$Y(k)+jY(N-k)=c(k)W_{4N}^{k}\sum_{n=0}^{N/4-1} d(n,k)W_N^{nk} \quad \text{, for k==0,2, ..., N12} \quad (17)$$

For the case that k is odd, we define a similar variable d'(n,k) by

$$d'(n,k)=v(2n)+j(-1)^{(k+1)/2}v(N/2-1-2n) \quad (18)$$

Then eqn(12) can be written as

$$Y(k)+jY(N-k)=c(k)W_{4N}^{k}\sum_{n=0}^{N/4-1} d'(n,k)W_N^{nk} \quad \text{, for k=1,3, ...,N/2-1} \quad (19)$$

Eqns(17) and (19) are the desired expressions for implementing DCT. But from eqn(17) we see that for the case k=O and k=N/2 the real part is equal to the imaginary part which leads to hardware redundancy. We,therefore, want to derive an expression for Y(0)+jY(N/2). We proceed as follows.

From eqn(17), we have

$$Y(0)=Re\left\{c(0)\sum_{n=0}^{N/4-1} d(n,0)\right\}$$

$$= \frac{1}{\sqrt{2}} \sum_{n=0}^{N/4-1} d(n,0) \tag{20}$$

$$Y(\frac{N}{2}) = Re\left\{ c(\frac{N}{2}) W_{4N}^{N/2} \sum_{n=0}^{N/4-1} d(n,N/2) W_N^{nN/2} \right\} = \frac{1}{\sqrt{2}} \sum_{n=0}^{N/4-1} d(n,0)(-1)^n \tag{21}$$

After some arrangements and assuming 8 is a factor of N, we have

$$Y(0)+jY(\frac{N}{2}) = \frac{1+j}{\sqrt{2}} \left[\sum_{n=0}^{N/8-1} [d(2n,0)-jd(2n+1,0)] \right]$$

(22)

Eqns(17),(19) and (22) are the final expressions used to implement 1-D DCT.

III.Flow-Graph and Comparison

Since complex multiplication is involved in our algorithm, we would like to adopt a trick [2] to reduce the number of multiplication from 4 to 3. Therefore the number of real multiplication in the 8x1 DCT is only 9 in our algorithm. The flow graph of the 8x1 DCT based on our algorithm is shown in Fig. 1.

In order to evaluate the performance of our algorithm several well-known DCT algorithms are compared. Table I and II list the number of multipliers and adders in a 1-D DCT. We can see that our method is quite good. It is only slightly inferior to the Hou's algorithm [8]. Although Hou's method has lower computation complexity, it generates only 2 outputs at a time, while ours is four. Therefore, our method is believed to be more competent for high speed computation.

N	4	8	16	32	64
chen [12]	6	16	44	116	292
Wang [14]	5	13	35	91	227
Lee [13]	4	12	32	80	192
Vetterli[10]		12	32	80	192
Suehiro [15]	4	12	32	80	192
Hou [16]	4	7	15	31	63
our method		*9*	*18*	*36*	*72*

Table I. Comparison of number of multipliers of 1-D DCT.

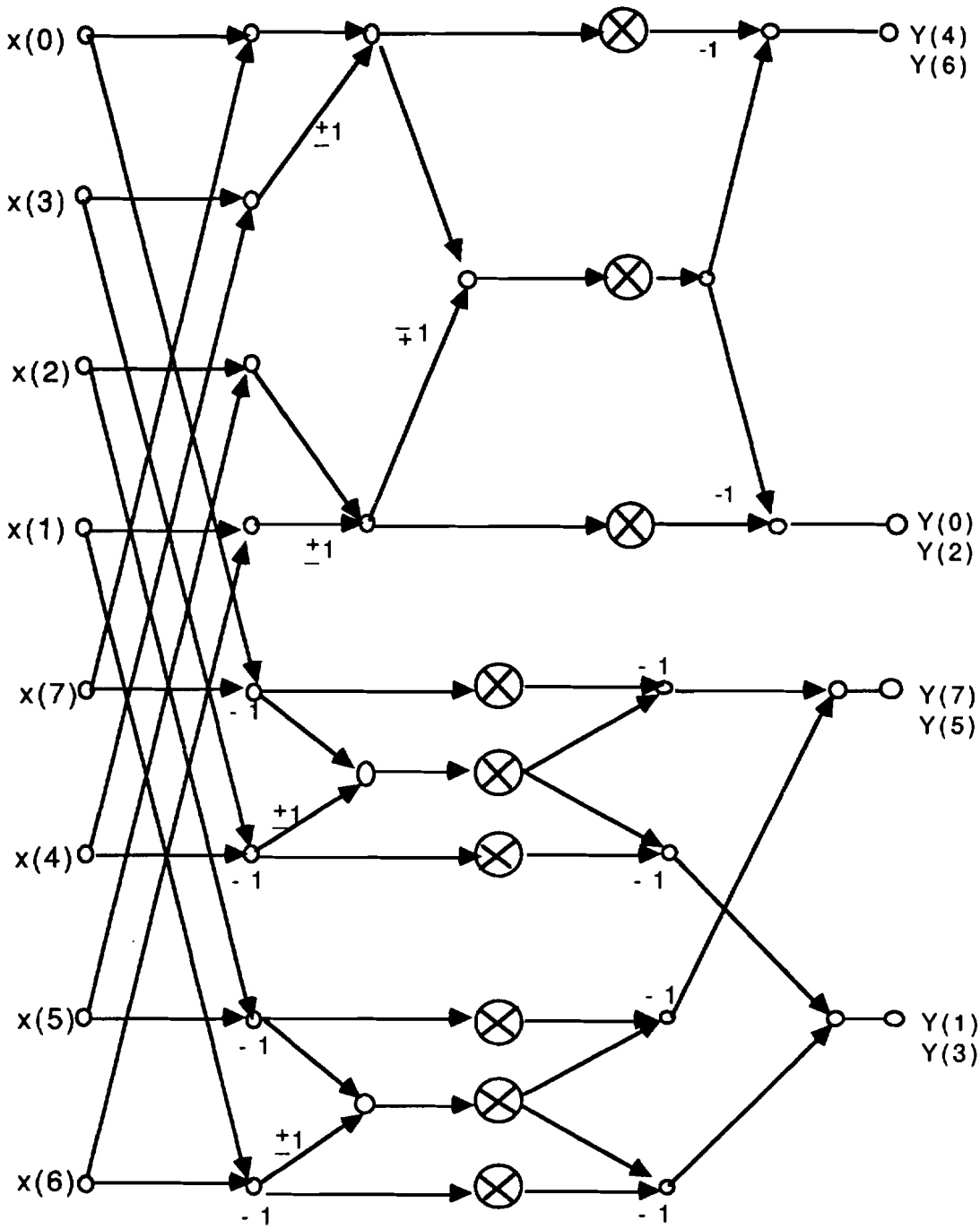


Fig.1. The flow graph of the 8x1 DCT based on the shared-multiplier algorithm.

N	4	8	16	32	64
chen [12]	8	26	74	194	482
Wang [14]	9	29	83	219	547
Lee [13]	9	29	81	209	513
Vetterli[10]		29	81	209	513
Suehiro [15]	9	29	81	209	513
Hou [16]	7	18	41	88	183
our method		21	48	100	204

Table II. Comparison of number of adders of 1-D DCT.

IV. Architecture

The 8x8 DCT can be implemented via 2 8x1 DCT and a transposition memory. The block diagram of the 8x8 DCT is depicted in Fig.2. Due to the limitation on pin numbers, the data sequence is word-parallelly sent into the input buffer of the front-end DCT. The word length of the input data is 8 bits; after the transformation, the computation results are coded with 12 bits , which meets the specifications of MPEG. The transformed sequence occurs also in natural order.

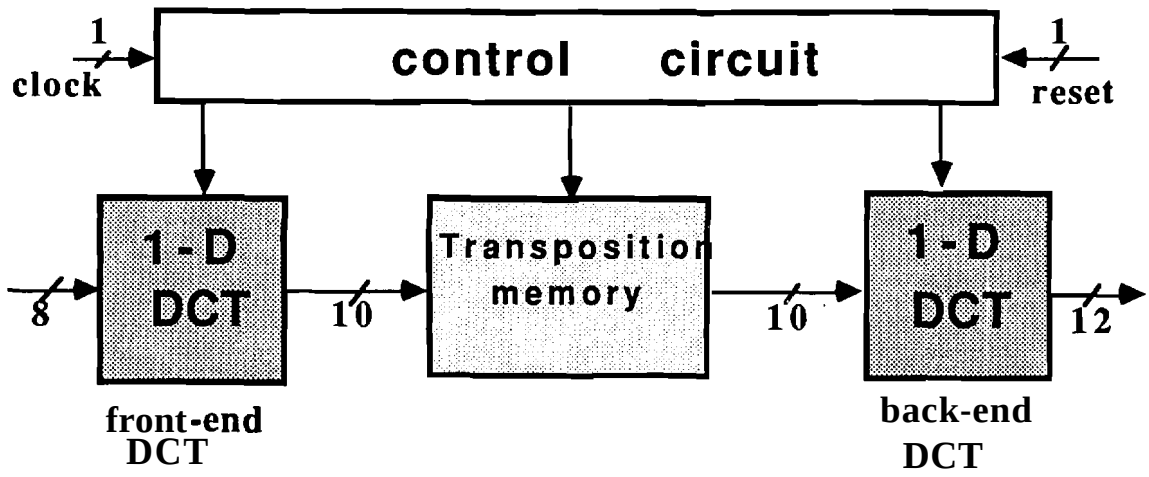


Fig.2. The block diagram of 2-D DCT.

The front-end DCT and back-end DCT in Fig.2 are almost the same. The differences lie in the word-length of their I/O and the design of their output buffers. Based on our algorithm, each 1-D DCT can be divided into several functional blocks shown below.

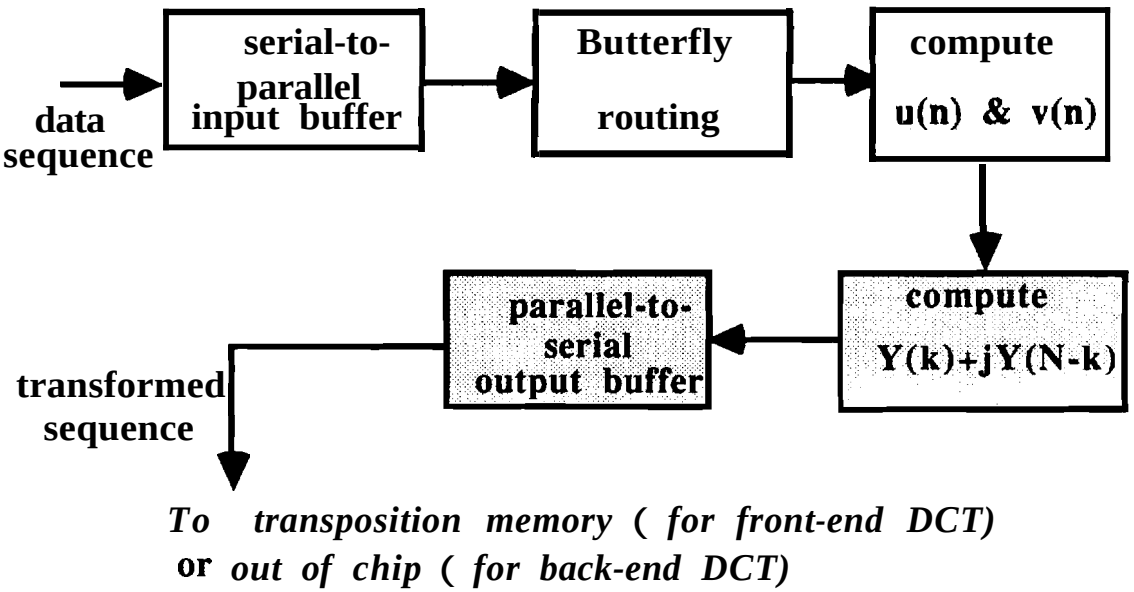


Fig.3. The functional blacks of 1-D DCT based on our algorithm.

Now we would like to explain the clacking strategy of the whole system. Please refer to Fig.4. The clock signal CK_1 is externally provided, while CK_4 and CK_8 are internally generated. CK_4 and CK_8 are used to pipeline the system, and CK_1 is synchronised with the input data. Each 8-point data block is computed within one CK_8 period. The complements of these clock signals are also generated by our circuit.

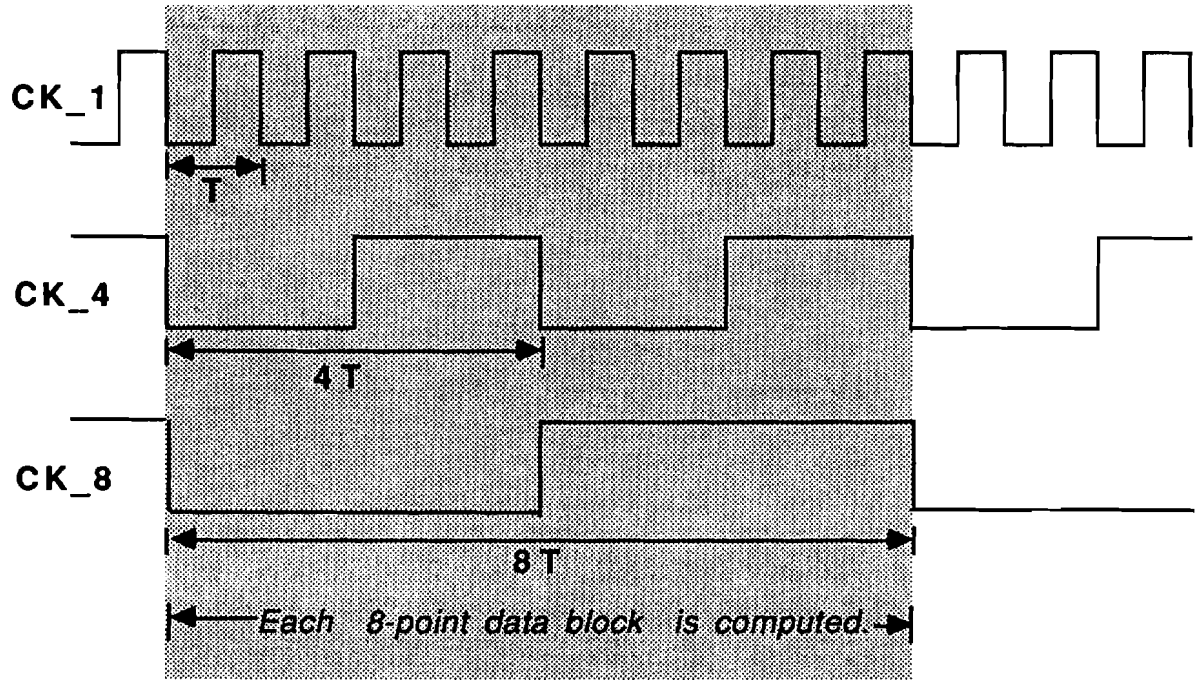


Fig.4. The clocks used in the system.

Let us go back to Fig.3. Since the input sequence appears in natural order and is serially fed into the system, a serial-to-parallel input buffer is therefore required to convert the input data to the parallel form. As shown in Fig.5., after the whole block (8 data) has entered the buffer, they are then parallelly shifted into shift registers. In Fig.6 we explain how to obtain the intermediate variables u(n) and v(n) form the input data. Since the additions and subtractions performed in this stage can be done within 8T which is well enough, the simple ripple-carry adders/subtractors are used in order to save area. With HSPICE simulation, the worst-case delay of the 8-bit adder is simulated to be less than 17ns. The 1I-bit adder is the largest adder in the whole system and its worst-case delay is simulated to be not longer than 22ns. In fact, throughout the system, all the additions and subtractions are allowed to be performed within 8T or 4T. Thus we do not intend to use fast adders/subtractors in our system.

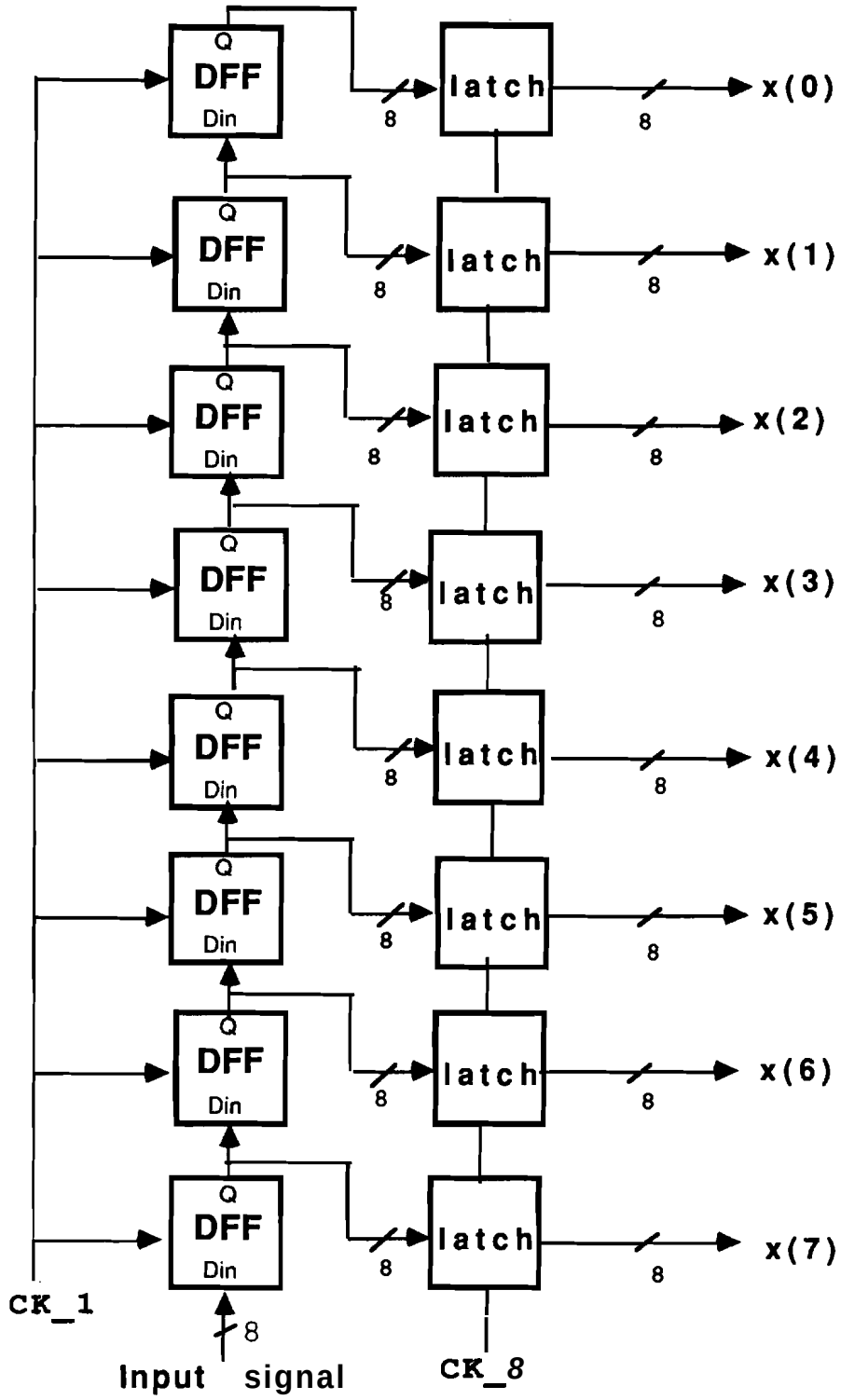


Fig. 5. The serial-to-parallel input buffer of the front-end DCT.

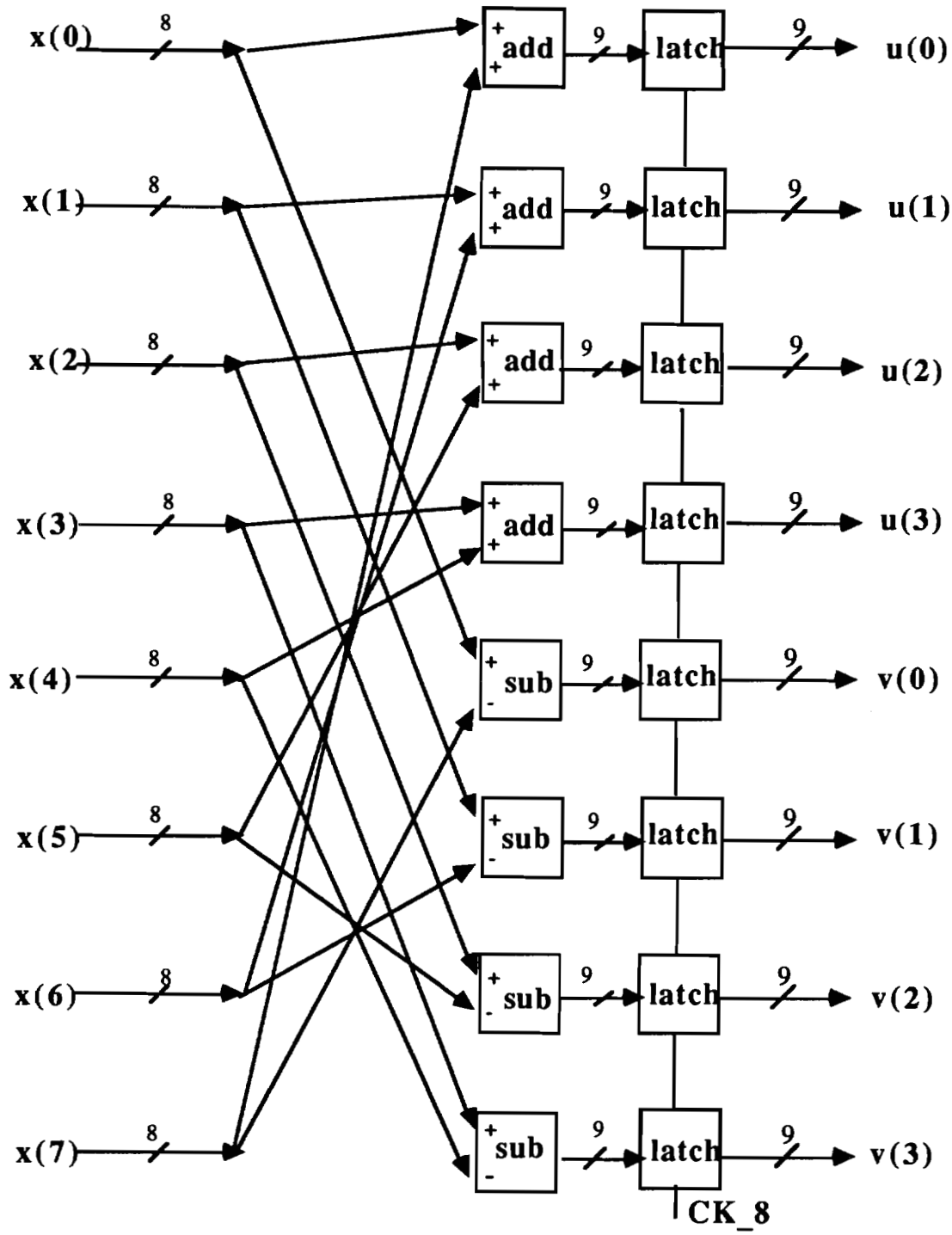


Fig.6. The Butterfly routing and adders/subtractors used to calculate u(n) and v(n).

From the u(n)'s in Fig.6 we can compute the even terms of the output Y(k)'s; The v(n)'s are then used to calculate the odd terms of the coefficients Y(k)'s. The block diagrams for finding even and odd terms of Y(k)'s are given in Fig.7 and Fig.8 respectively. Note that the values listed in the blocks of multipliers are the values of $\cos\theta+\sin\theta$, $\cos\theta$ and $\cos\theta-\sin\theta$ where θ corresponds to an angle associating with a certain n. If the control signal is 0/1, the operation or the value above/under the slash is involved in the computation. Thus, we obtain Y(0),Y(4),Y(1) and Y(7) during the former 4T; while Y(2),Y(6),Y(3) and Y(5) are calculated during the latter half of a data period.

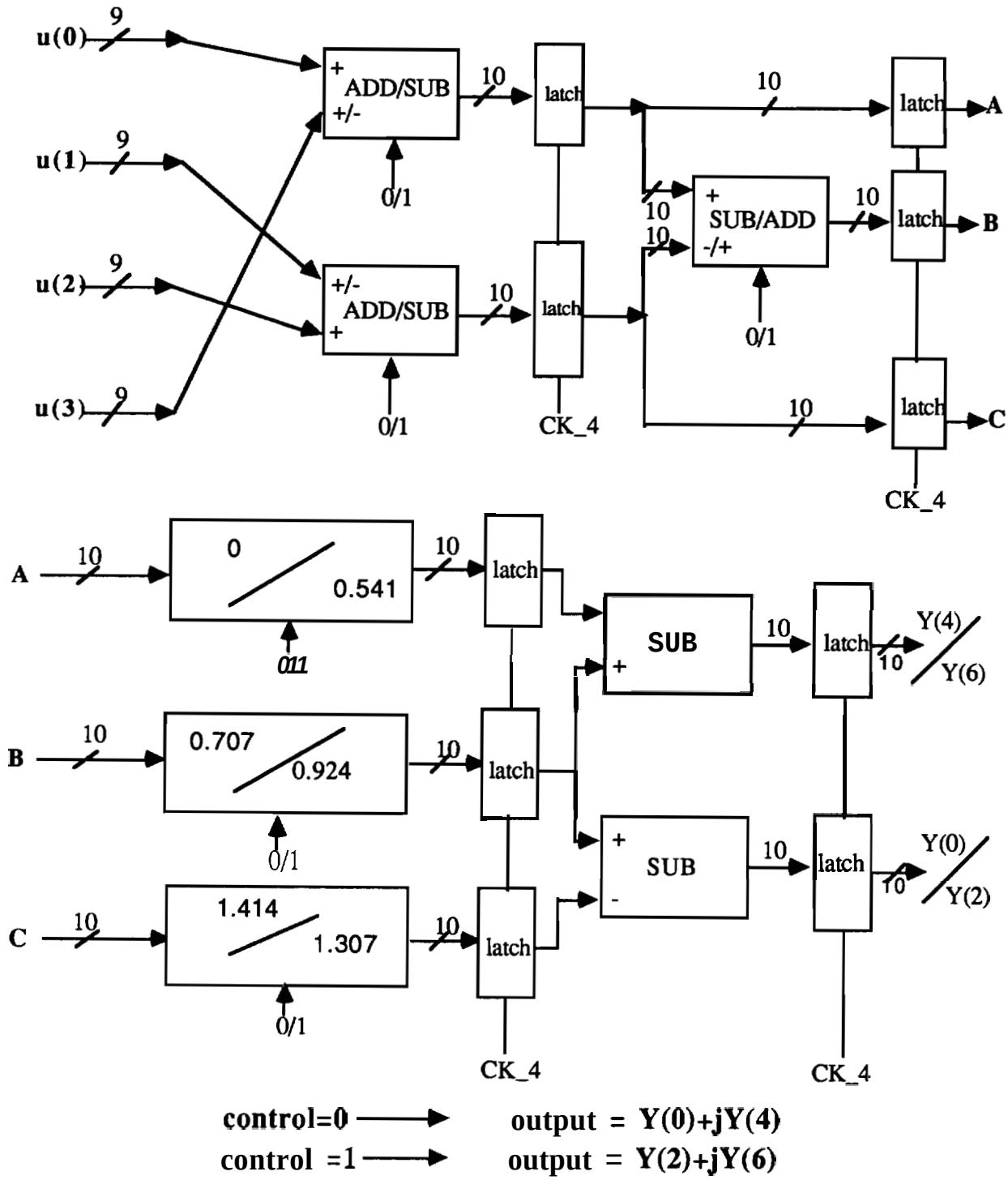
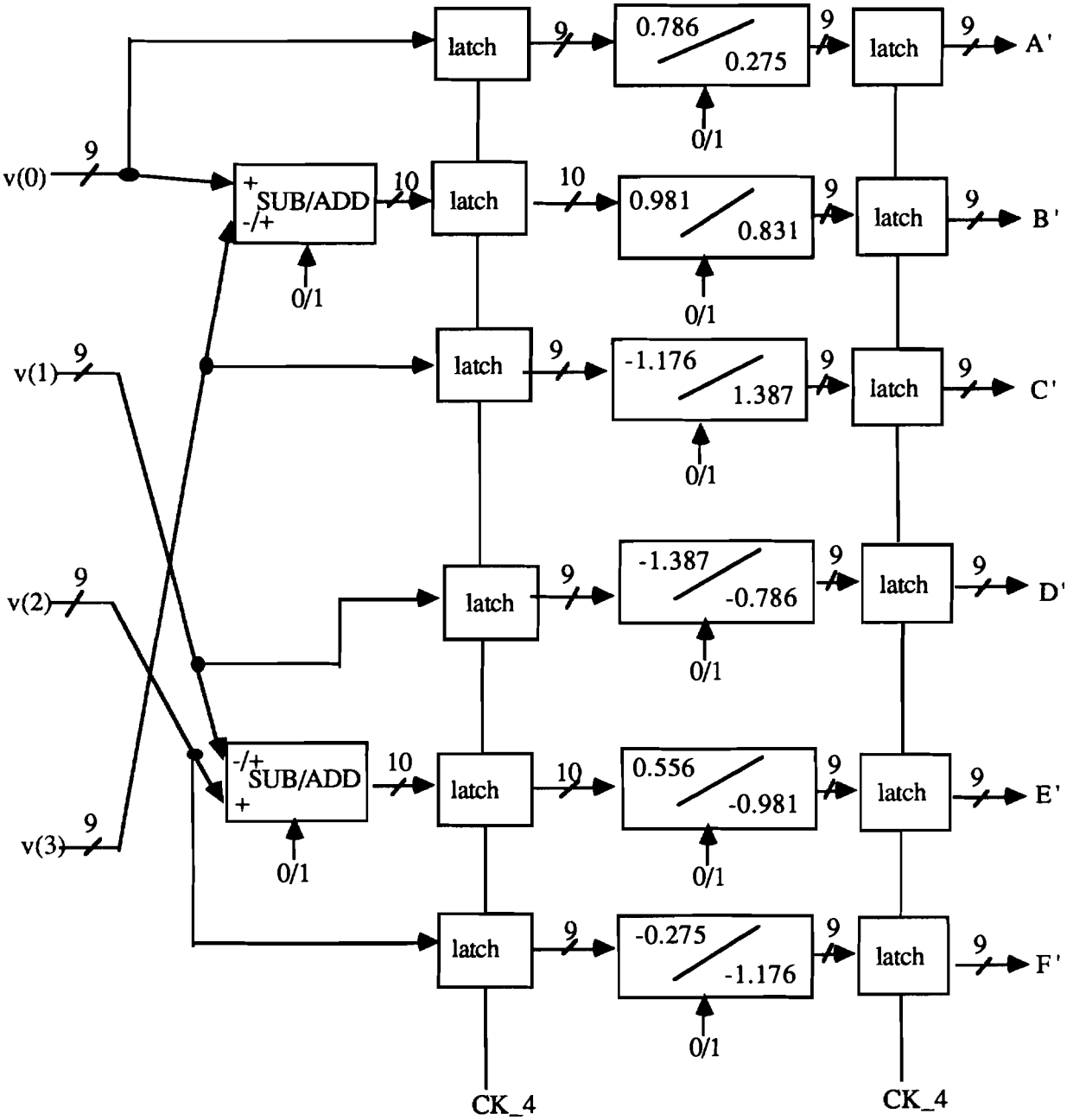


Fig.7. The block diagram for calculating the even part of output.


v(0)
v(1)
v(2)
v(3)
9
+
SUB/ADD
-/+
10
0/1
latch
0.786
0.275
0.981
0.831
-1.176
1.387
-1.387
-0.786
0.556
-0.981
-0.275
-1.176
A'
B'
C'
D'
E'
F'
CK_4

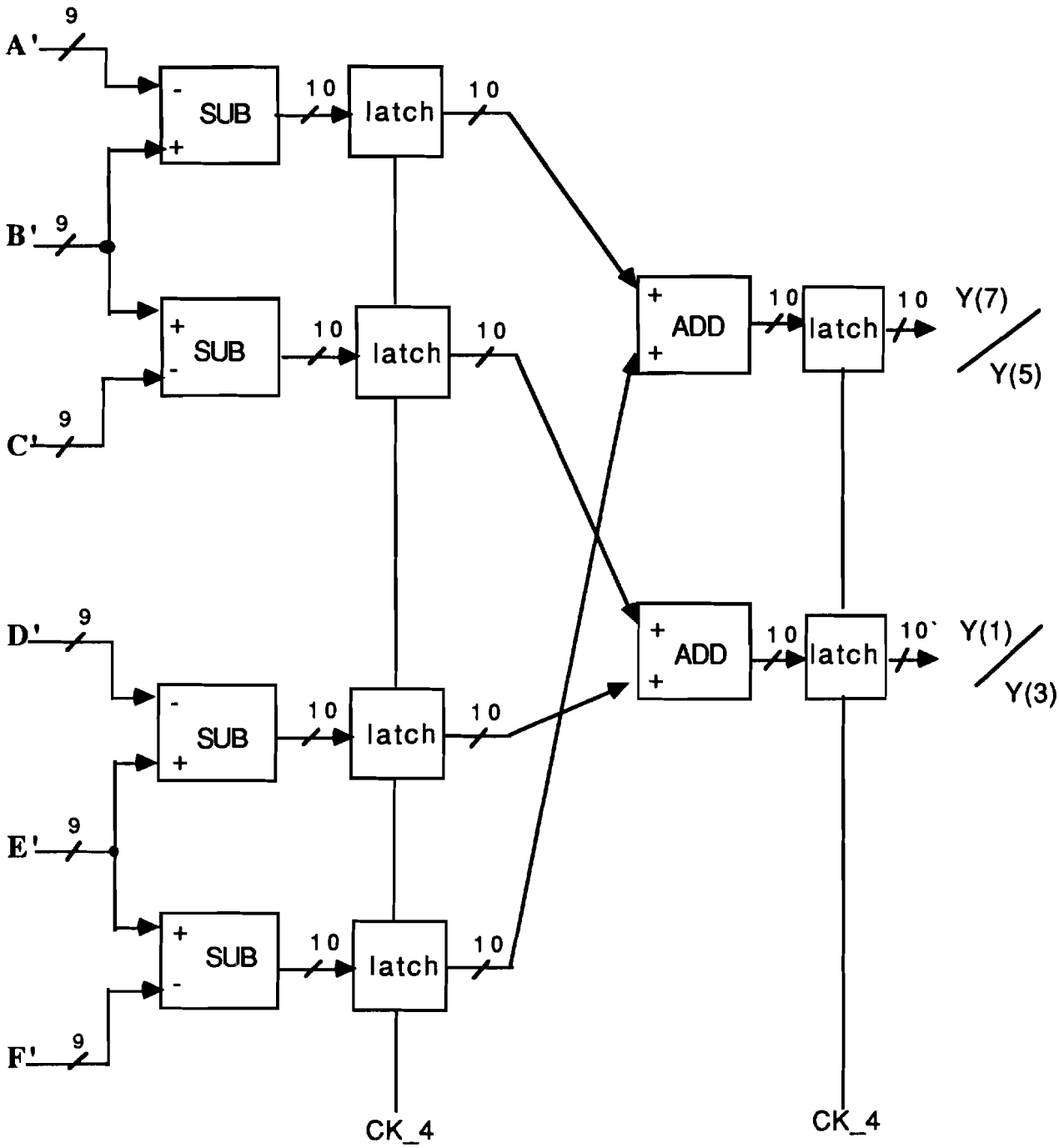


Fig.8. The block for finding the odd part of the output.

The designs of the output buffer for the front-end DCT and back-end DCT are quite different. Since the data written from the front-end DCT to the transposition memory can be in decimated order, the output buffer of the front-end DCT is much simpler. It is merely a multiplexer which arbitrates the data to appear in the order of Y(0),Y(4),Y(1),Y(7),Y(2),Y(6),Y(3),Y(5) for each data cycle as shown in Fig.9.

The occurring order of Y(k) :

T0	T1	T2	T3	T4	T5	T6	T7
Y(0)	Y(4)	Y(1)	Y(7)	Y(2)	Y(6)	Y(3)	Y(5)

The lowest 2 bits of the 7-bit RAM counter

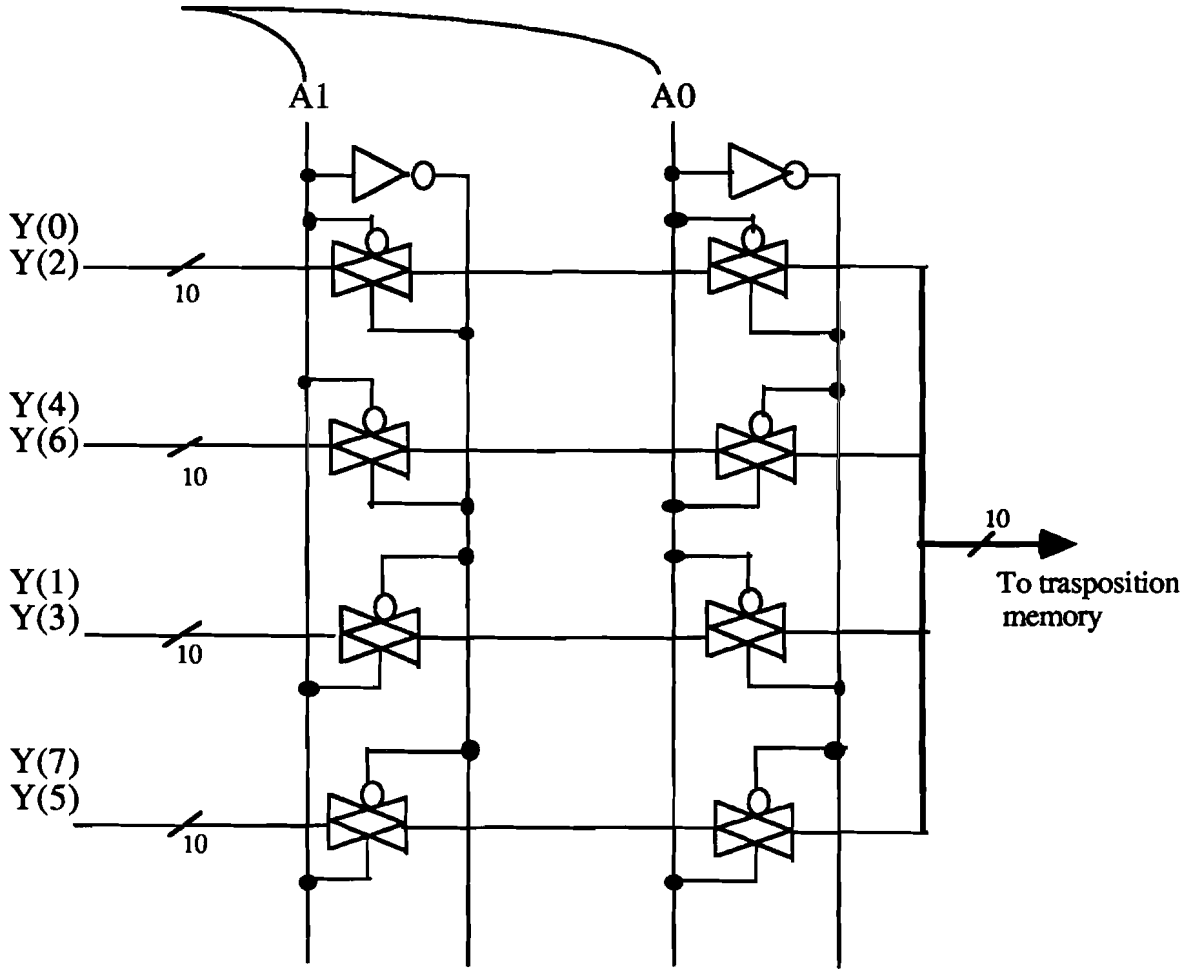
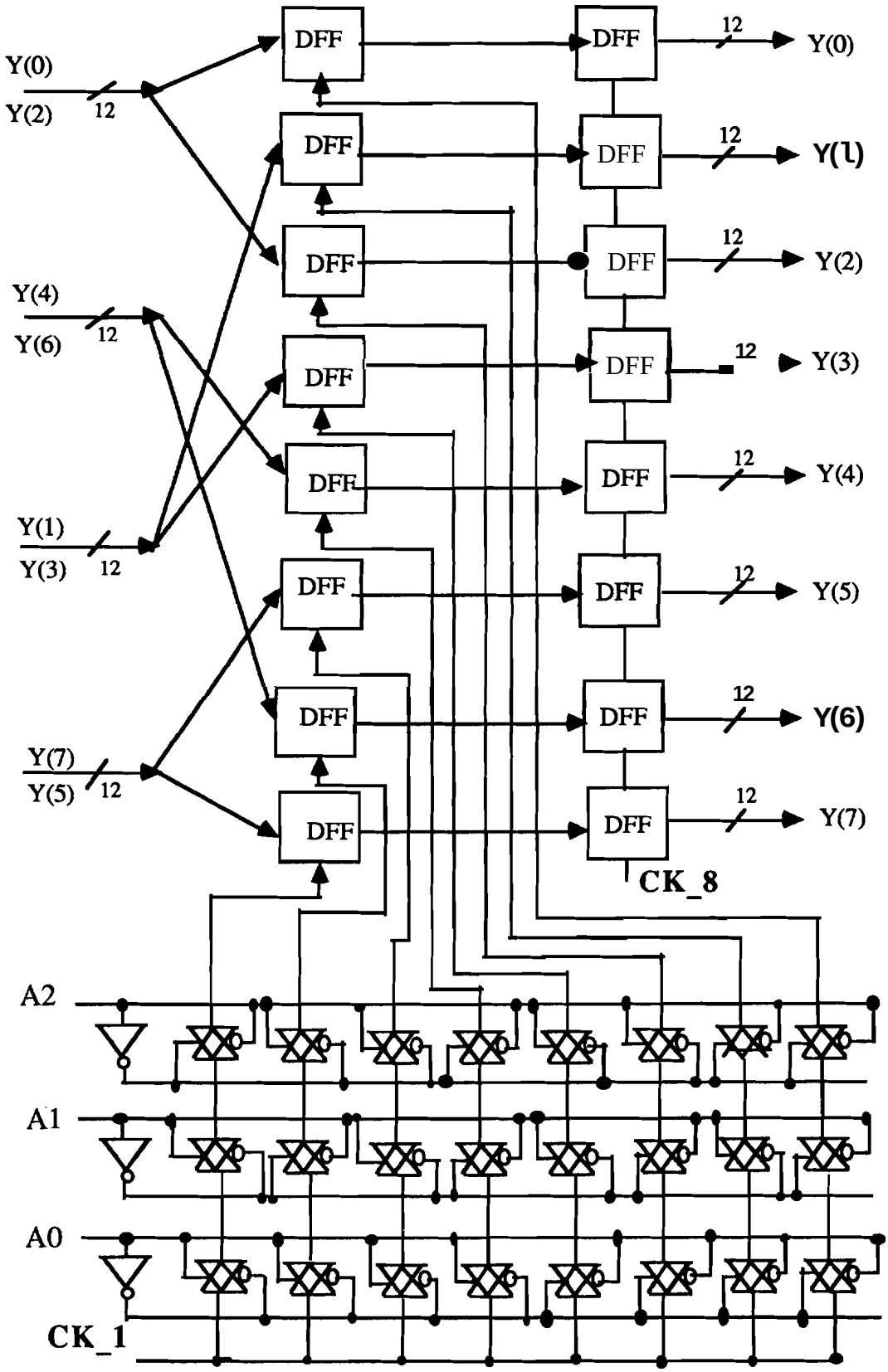


Fig.9. The parallel-to-serial output buffer of the front-end DCT.

The output buffer of the back-end **DCT** is much more complex than that of the front-end because the data from the back-end **DCT** *are* to be sent out of the chip. They, therefore, have to appear in the natural order. As shown in Fig.10. we first extend the data into fully parallel form which is done

by controlling the clock signal to the DFF via a demultiplexer. The clock signal CK_1 is connected to one and only one DFF for each T cycle. After that all 8 data have arrived, they are then chosen by a multiplexer to occur from k=0 to 7.


Y(0)
Y(2)
12
Y(4)
Y(6)
12
Y(1)
Y(3)
12
Y(7)
Y(5)
12
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
DFF
12
Y(0)
12
Y(1)
12
Y(2)
12
Y(3)
12
Y(4)
12
Y(5)
12
Y(6)
12
Y(7)
CK_8
A2
A1
A0
CK_1

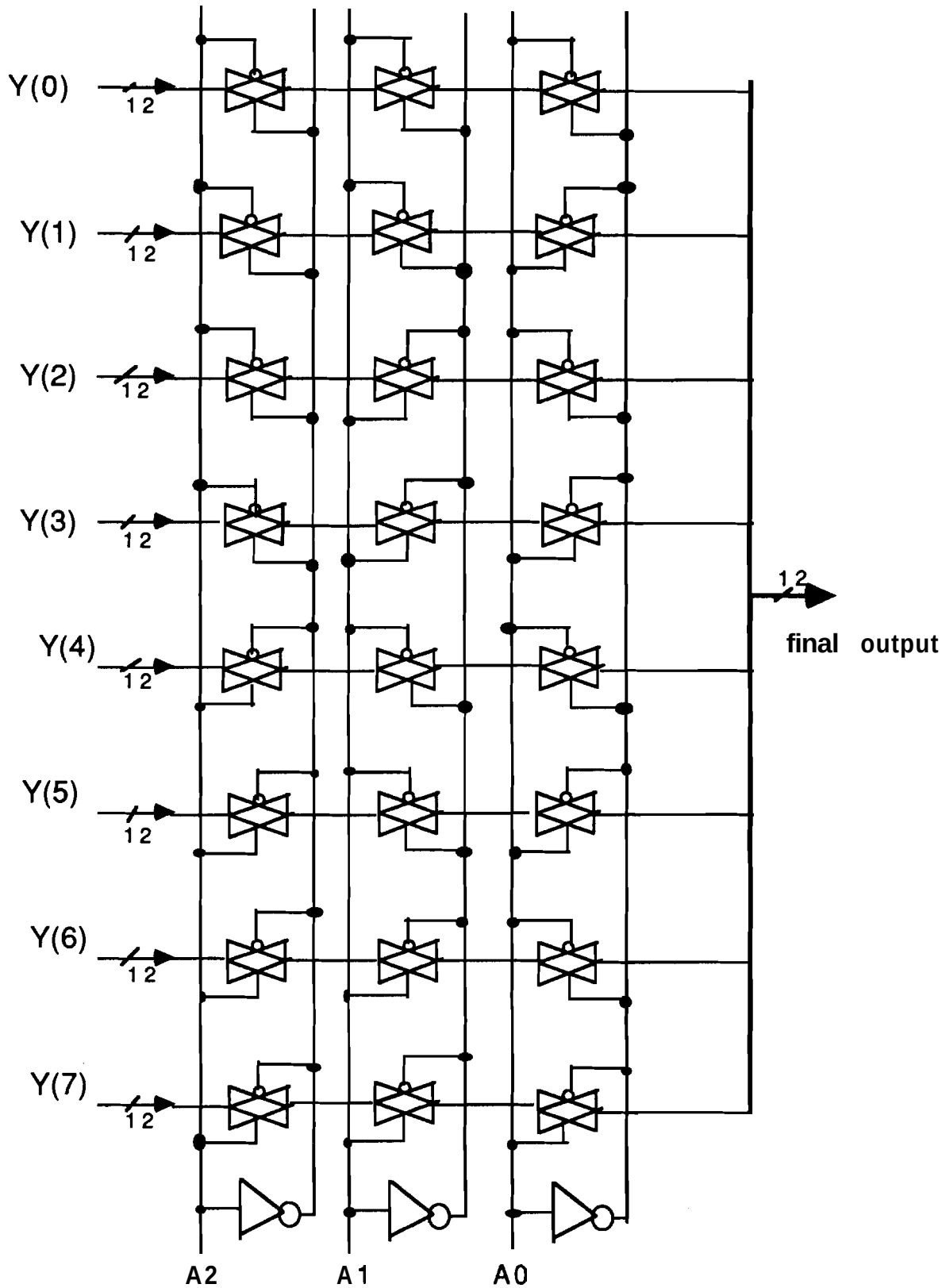


Fig.10. The output buffer of the back-end DCT.

V. Hardware Implementation

From the HSPICE simulation results we know that the critical path in our system is the 2's complement multiplier. As one can see in Fig.7 and Fig.8 that an interval of 4T is allowed to perform a multiplication, the multiplier itself need not to be very fast ,but its area must be as small as possible. Our multiplier is based on the modified Booth algorithm with the area of approximately 1mm x 0.85mm. It is simulated to have the worst case delay of 40ns, i.e. 4T=40ns. We thus have T=10ns. This means that our DCT chip can operate at the maximum speed of 100MHz. The layout of the whole system is shown in Fig.11. The layout is drawn by way of fully custom design with the 1.2um double-metal single-poly CMOS technoology. The area is 8.6mm x 8.8mm. The total pin number is 34. The function of the whole system, including the transposition, has been verified with IRSIM (a layout simulation tool) to be correct.

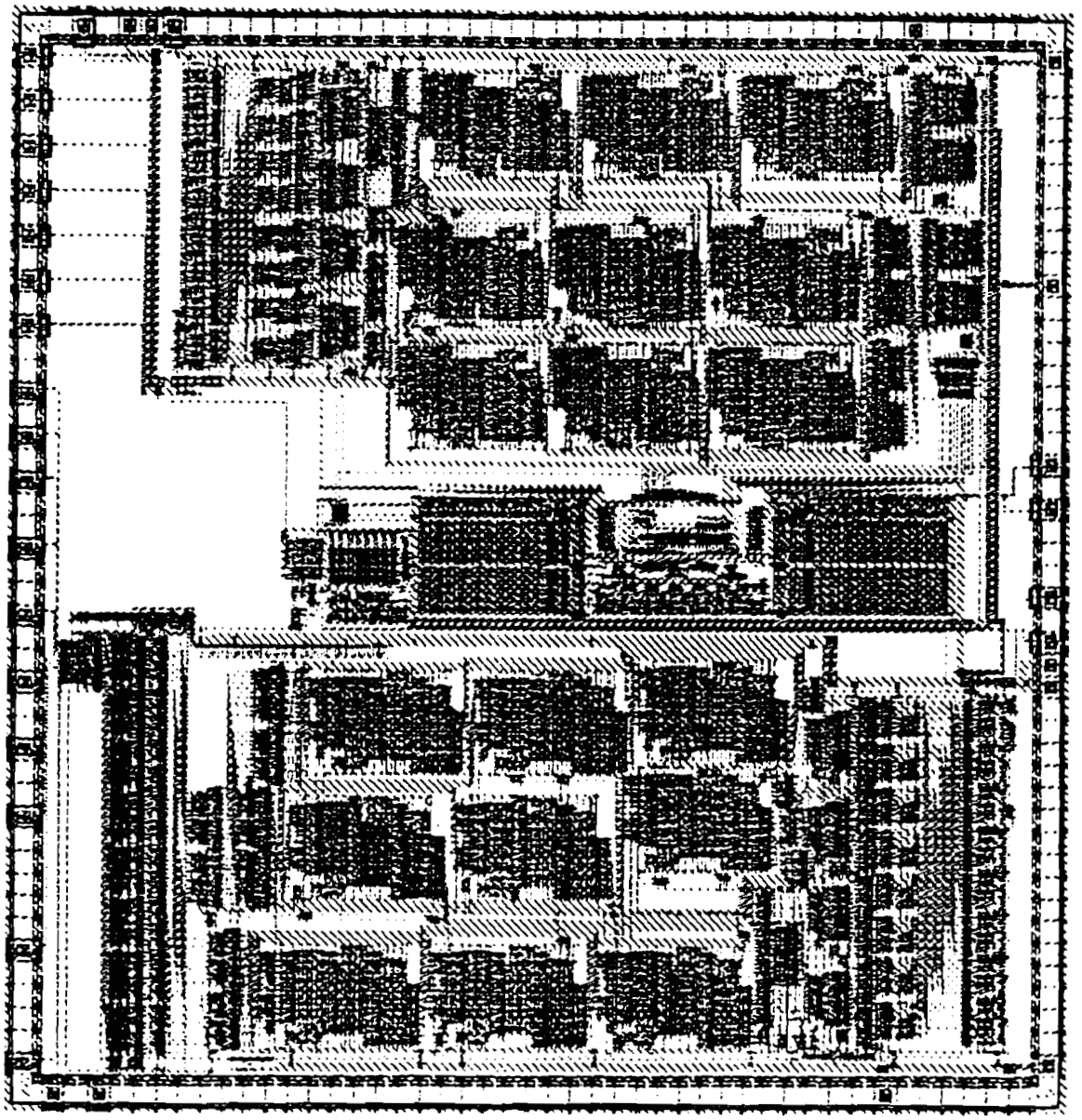


Fig.11. The layout of the 8x8 DCT based on the shared-multiplier algorithm.

VI. Conclusions

This chip is a prototype which proves that it is feasible to implement the 8x8 DCT on a single chip with data rate over the speed requirement of the HDTV system. Two major improvements can be made to reduce the circuit area significantly. First, better floor planning may reduce the circuit area by 10%. Both the routing and space area can be reduced Second, refining the design of the multiplier can reduce the circuit area by another 10%. With these impromentsmentioned above, the core area can approximately be shrunk to only 7mm x 7mm.

The sharing of multipliers in computing DCT is a brand new attemp and it is proven to work efficient, even better than our expectation. If some application requires the data rate to be higher than 100MHz, one simly has to enhance the speed of the multipliers. If the delay time of the multiplier is reduced from 40ns to 30ns which does not increase the circuit area very much, the overall data rate can be increased from 100MHz to 133MHz.