Design of A 100MHz 64-Point FFT Processor in 0.35µm Standard CMOS Technology

HOJAT MOJARAD, HASSAN HAJGHASSEM
Department of electrical engineering
Shahid Beheshti University
Tehran
IRAN
Hojat_mojarad_alman@yahoo.com

Abstract: applications such as digital spectrum analyzers, digital filtering, image processing, and video transmission need to compute the Discrete Fourier Transform (DFT). A Chip architecture to compute a 64-point DFT using radix-4 algorithm every 18.87 µs at 100MHz clock rate is designed. This processor incorporates static memory, controller, and combinational operating unit (COU). Input data and output data are 10-bit and 54-bit words, respectively. A 10-bit × 27-bit multiplier is used inside the processor. Two's complement is used to present negative data. This processor is implemented in 0.35µm tsmc standard CMOS process.

Key-Words: DFT, decimation in time, radix-4, COU, complex multiplier, decimation in frequency

1 Introduction

To perform frequency analysis on a discrete-time signal {x(n)}, we convert the time-domain sequence to an equivalent frequency-domain representation. Such a representation is given by the Fourier transform of the sequence {x(n)}. In particular, important computational algorithms, called fast Fourier transform (FFT), is presented for computing the DFT when the size N is a power of 4.

Basically, DFT is to compute a sequence {X(K)} of N complex-valued numbers, given another sequence of data {x(n)} of length N, according to the formula :

$$X(K) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad 0 \le k \le N-1 \qquad (1)$$

Where

$$W_N = e^{-j2\pi/N} \qquad (2)$$

In general, the data sequence x (n) is also assumed to be complex valued.

We observe that for each value of k, direct computation of X (K) involves N complex multiplications (4N real multiplications) and N-1 complex additions (4N-2 real additions).
Consequently, to compute all N values of the DFT, we need N^2 complex multiplication and N^2-N complex additions.
Direct computation of the DFT is basically inefficient, because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular these properties are:

Symmetry: $$W_N^{k+N/2} = -W_N^k \qquad (3)$$

Periodicity: $$W_N^{k+N} = W_N^k \qquad (4)$$

The computationally efficient algorithms known as FFT algorithms, exploit these two basic properties of the phase factor.

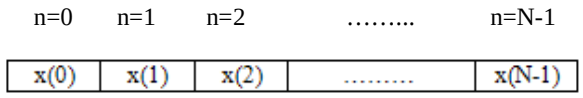
2 Divide-and-Conquer Approach

This approach is based on the decomposition of an N-point DFT into successively smaller DFT's. To illustrate the basic notations consider the computation of an N-point DFT, where N can be factored as a product of two integers, that is,

$$N=LM \qquad (5)$$

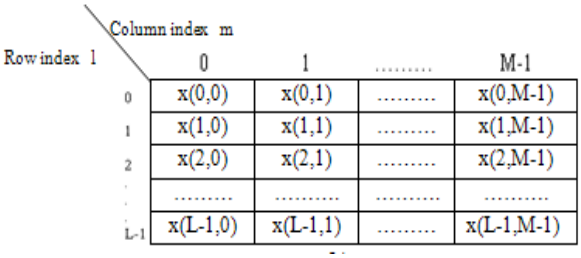
The assumption that N is not a prime number is not restrictive, since we can pad any sequence with zeros to ensure a factorization of the form mentioned.

Now the sequence x (n), 0≤ n ≤ N-1 can be stored in either a one-dimensional array indexed by n or as a two-dimensional array indexed by l and m, where 0≤ l ≤L-1 and 0≤ m ≤M-1 as illustrated in Fig.1.



(a)

Fig.1.a. One-directional array



(b)

Fig.1.b. Two-dimensional array

Thus the sequence x(n) can be stored in a rectangular array in a variety of ways each of which depends on the mapping of index n to the indexes (l,m).

A similar arrangement can be used to store the computed DFT values. In particular, the mapping is from the index k to a pair of indices (p,q), where 0≤ p≤ L-1 and 0≤q≤M-1. If we select the mapping:

k=Mp+q (6)

The DFT is stored on a row-wise basis, where the first row contains the first M elements of the DFT X(K), the second row contains the next set of M elements, and so on. On the other hand, the mapping:

k=qL+p (7)

results in a column wise storage of X(K). Now suppose that x (n) is mapped into the rectangular array x (l, m) and X (K) is mapped into a corresponding rectangular array X (p, q). Thus the DFT can be expressed as a double sum over the elements of rectangular array multiplied by the corresponding phase factors. Then: X (p, q) = $\sum_{m=0}^{M-1}\sum_{l=0}^{L-1} x(l,m)W_N^{(Mp+q)(mL+l)}$ (8)

But

$W_N^{(Mp+q)(mL+l)} = W_N^{MLmp}W_N^{mLq}W_N^{Mpl}W_N^{lq}$ (9)

However, $W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$, and $W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$.

With these simplification (8) can be expressed as:

$$X(p,q) = \sum_{l=0}^{L-1}\{W_N^{lq}\,[\sum_{m=0}^{M-1} x(l,m)W_M^{mq}]\}W_L^{lp} \qquad (10)$$

The expression (10) involves the computation of DFT's of length M and length L. to elaborate, let us subdivide the computation into three steps:

1. First, we compute the M-point DFT's:

$$F(l,q) = \sum_{m=0}^{M-1} x(l,m)W_M^{mq} \quad 0 \le q \le M-1 \qquad (11)$$

For each of the rows l=0, 1, 2,……., L-1.

2. Second, we compute a new rectangular array G(l, q) defined as:

$$G(l,q) = W_N^{lq}F(l,q) \quad 0 \le l \le L-1 \qquad (12)$$

$$0 \le q \le M-1$$

3. Finally, we compute the L-point DFT's

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q)W_L^{lp} \qquad (13)$$

For each column q=0, 1, 2, ….., M-1 of the array G (l, q).

First it seems that the computational procedure described above is more complex than direct computation of the DFT. However, let us evaluate the complexity of (10). The first step involves the computation of L DFT's, each of M points. Hence this step requires LM^2 complex multiplication and LM(M-1) complex additions. The second step requires LM complex multiplications and third step requires ML^2 complex multiplications and ML(L-1) complex additions. Therefore the total number of computational complexity is:

Complex multiplications: N (M+L+1)

Complex additions: N (M+L-2)

Where N=LM. Thus the number of multiplications has been reduced from N^2 to N(M+L+1) and the number of complex additions has been reduced from N (N-1) to N (M+L-2). In the following section the divide-and-conquer approach is exploited to derive fast algorithms when the size N is a power of 4.

3 Radix-4 Algorithm

When the number of data points N in the DFT is a power of 4, we can use radix-4 algorithm for computations which is more efficient.

Let us begin by describing a radix-4 decimation-in-time FFT algorithm, which is obtained by selecting L=4 and M=N/4 in the divide-and-conquer approach described in previous section. For this choice of L and M, we have l, p= 0, 1, 2, 3 and m, q=0, 1, 2,…,(N/4)-1 and n=4m+l and k=(N/4) p+q. Thus we split or decimate the N- point input sequence into four sub sequences, x(4n), x(4n+1), x(4n+2), x(4n+3), n=0, 1, 2,…,(N/4)-1.

By applying (10):

$$X(p,q) = \sum_{l=0}^{3}[W_N^{lq}F(l,q)]W_4^{lp}, \quad p = 0,1,2,3 \qquad (14)$$

Where $F(l,q)$ is given by (11), that is:

$$F(l,q) = \sum_{m=0}^{\left(\frac{N}{4}\right)-1} x(l,m)W_{\frac{N}{4}}^{mq} \quad l = 0,1,2,3 \qquad (15)$$

$$q = 0,1,1,\dots,\frac{N}{4}-1$$

Note that $x(l,m) = x(4m+l)$ and $X(p,q) = X(\frac{N}{4}p+q)$.

Thus the four N/4-point DFT's obtained from (15) are combined according to (14) to yield the N-point DFT. The equation (14) for combining the N/4-point DFT's defines a radix-4 decimation-in-time butterfly, which can be expressed in matrix form as:

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^{\,0}F(0,q) \\ W_N^{\,q}F(1,q) \\ W_N^{\,2q}F(2,q) \\ W_N^{\,3q}F(3,q) \end{bmatrix} \qquad (16)$$

The radix-4 butterfly is shown in Fig.2. We can see that each butterfly involves three complex multiplications, since $W_N^{\,0} = 1$, and 12 complex additions.

The decimation-in-time procedure described earlier can be repeated recursively υ times where $N=2^{\upsilon}$ and each stage contains N/4 butterflies. As a result the total number of multiplications is reduced by 25% and also the number of additions is reduced by 25%.

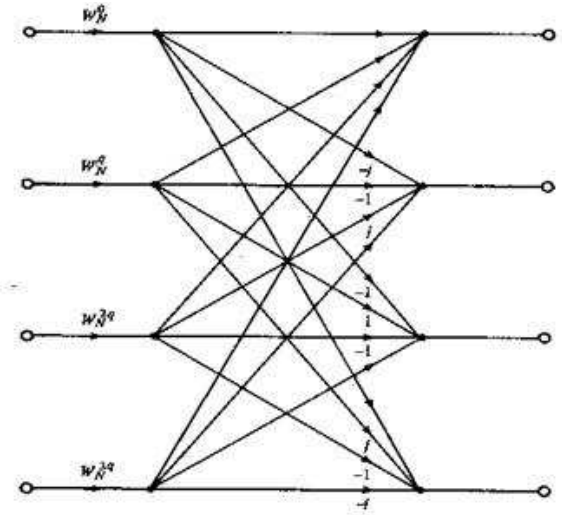


Fig.2. Basic butterfly computation in radix-4

4 System Structure

Among the various algorithms, radix-4 FFT algorithm has the least multiplication operations. Moreover, needs less cascaded stages in comparison to the other algorithms, like radix-2. For example four stages are needed for radix-2, while only two stages are required by radix-4 for a 16-point DFT. Thus we use radix-4 in our design.

Based on the basic butterfly in the radix-4, we designed a processor which can compute the 64-point DFT. The block diagram of processor is shown in Fig.3. General operation of the processor is as follows:

1. To write the input data to the memory
2. To read the data in sets of four at a time, from memory and compute their 4-point DFT.
3. To write the results to the memory.
4. To repeat steps 2 and 3, 48 times.
5. To read the output data from memory.

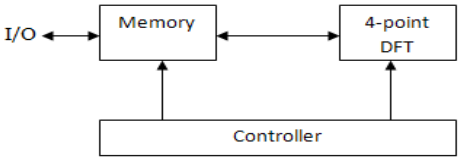


Fig.3. 64-point DFT processor architecture

4.1 Memory

Two types of memory, RAM and ROM, are used in the 64-point DFT processor. RAM is used for input data, to store intermediate computations and to output data. ROM is used as the phase factor table.

4.1.1 RAM

Chip input data are binary 10-bit words. In other words, the value of x(n) are 10 bit words. For N=64

$$X(K) = \sum_{n=0}^{63} x(n) W_{64}^{kn} \qquad 0 \le k \le 63 \qquad (17)$$

for k=0 we have : $X(0) = \sum_{n=0}^{63} x(n)$ (18)

Assume that all the values of 64 data for x(n) are maximum, it means x(n)=$(11111\ 11111)_2$ where $0 \le n \le 63$. Under these circumstances, summation result in (18) is a 16 bit word. Another bit is used for sign and makes 17 bit word. Also, since values of phase factors W_{64}^{kn} are between -1 and 1, therefore summation terms have both integer part and real part. We add 10 bits to word for expressing the decimal part. For hardware implementation we used static RAM that is shown in Fig.4.

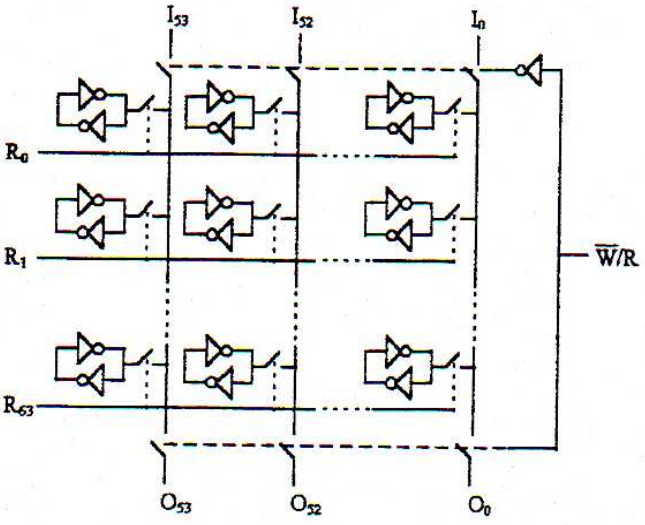


Fig.4.Schematic representation of RAM

4.1.2 ROM

The phase factors for 64-point DFT are constant. For this reason they are stored in a ROM. Each phase factor is multiplied by the 4-point DFT processor input data.

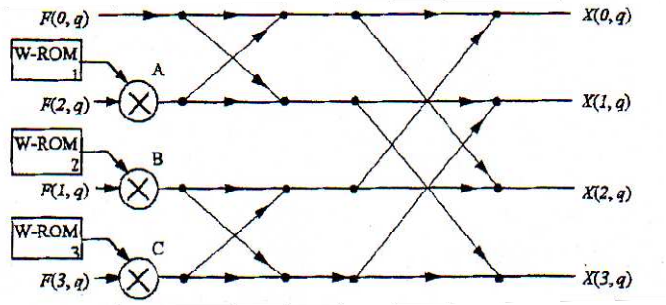


Fig.5. W-ROM's position in the 4-point DFT processor

Each of W-ROMs has 16 words and we choose 22bits for every word, 11 bits for real part and 11 bits for imaginary parts.

4.2 4-Point DFT Processor

The 4-point DFT processor consists of three main parts: the complex multipliers, the combinational operating unit (COU), and the latches. Fig.6. illustrates the structure of the 4-point DFT processor where A and B shows latches.

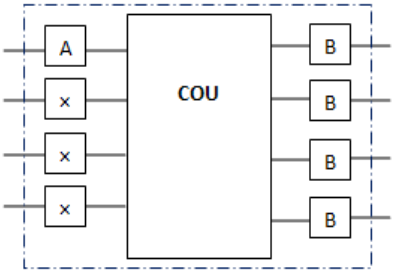


Fig.6. 4-point DFT processor structure

4.2.1 Combinational Operating Unit

COU is the heart of 4-point DFT processor. Its task is to realize the following operation:

$$X(K) = \sum_{n=0}^{3} x(n) e^{-j2\pi kn/4} \qquad (19)$$

A primary COU describing this equation is shown in Fig.7., where the notation of -1 indicates two's complement and block of –j indicates -j multiplier.

Therefore this unit consists of sixteen 27 bit adders and eight two's complement circuit.

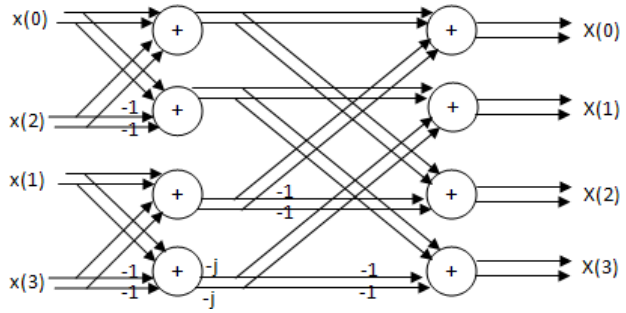


Fig.7. COU of a 4-point DFT processor

4.2.2 Adder

We have used eight complex adders in the 4-point DFT processor. Each complex adder is composed of two 27 bit parallel adders.

Among different configurations, complementary pass transistor logic family provides compact silicon area and high speed compared to conventional CMOS logic. A full adder implementation is shown in Fig.8.

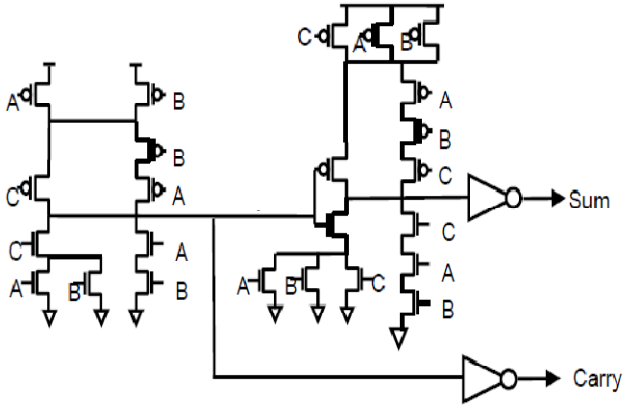


Fig.8. CPL full adder

A 27-bit parallel adder is indicated in Fig.9. Each digit except LSB requires a full adder. Since full adders are cascaded, there is a maximum propagation delay when A= " 1111111111" and B varies from "0000000000" to " 0000000001".

Obtained propagation delay is 9.2ns.

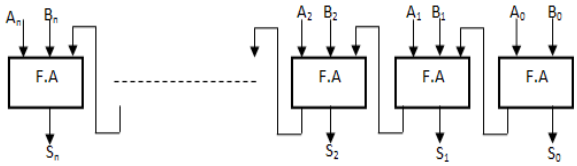


Fig.9. A 27 bit parallel binary adder

4.2.3 Two's Complement Circuit

To subtract one digital word from another one, we add the minuend to the two's complement of subtrahend. If there is an end carry out of the sign digit, we ignore it. Two's complement circuit is shown in Fig.10.

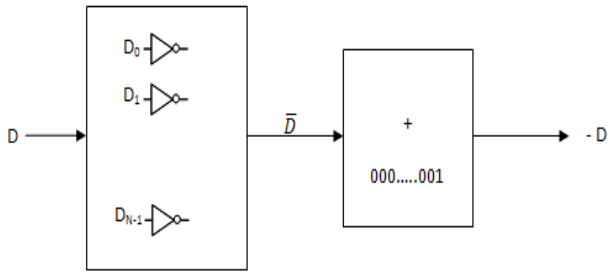


Fig.10. two's complement circuit

4.2.4 j and –j multiplier

These blocks can multiply a complex number by j or –j. suppose that the complex number has a real part A and imaginary part B. when it is multiplied by –j we obtain

$$(A + jB)(-j) = B - jA \qquad (20)$$

from this equation we observe that multiplying a complex number by –j results two's complement of A and exchanging the real part with imaginary part as shown in Fig11.. Also for j multiplier the same routine is repeated.

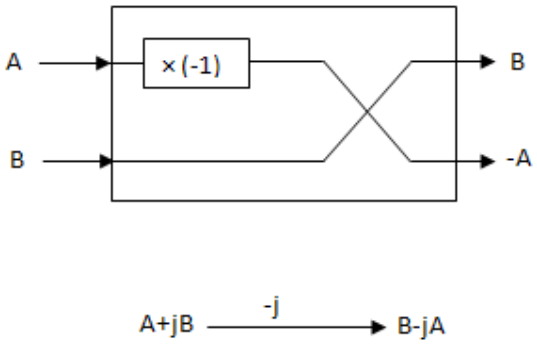


Fig.11. multiplication of a complex number by -j

4.2.5 Complex Multiplier

One of the most important parts of the processor is the complex multiplier. We mentioned that W-ROM stores phase factors. Both phase factors and input data are complex numbers, which are multiplied as in Fig.12.

Below we discuss multiplication operation and then present the 10-bit × 27-bit multiplier which is used in the processor.

Multiplication is a repeated process of left-shift and adds operations. Starting from the LSB, if it is 1, the multiplicand is copied to form the first partial product. If it is zero, an all-zero sequence forms the first partial product. This process continues until all the bits of the multiplier are exhausted. All the partial products are then summed to form the final product. The sign of product is determined from the signs of the multiplicand and multiplier.

Using this definition, a simple multiplier can be designed as shown in Fig.12 in which the multiplicand is stored in register M, the multiplier in shift register N, and the product in shift register Q.

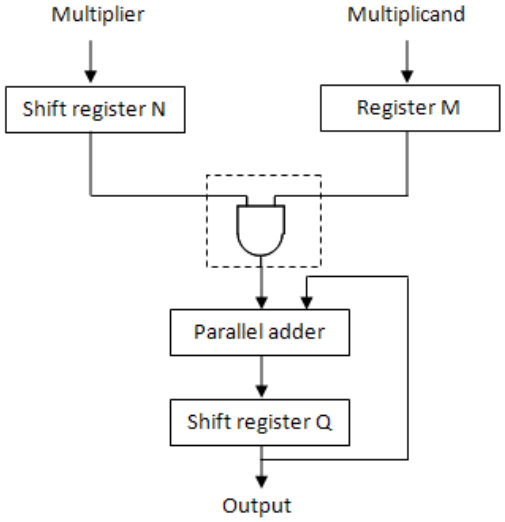


Fig.12.multiplier block diagram

4.2.6 Latches

As we previously mentioned, we have used two kinds of latches in 4-point DFT processor. Moreover, another type of latch was used in chip input for the purpose of latching the input and output data. To illustrate these latches, let us consider more carefully the structure of 4-point DFT processor in fig.13.

Latch groups of IL, ML, NL, and OL cause the processor to operate in pipeline. First, four complex data from the static memory are loaded into the latch group of IL. Then these four complex data are loaded into the latch group of ML and simultaneously second four complex data are loaded into the latch group of IL. The latches of ML1, ML2 and ML3 are in fact the same 26-bit input shift registers of the multipliers. Moreover, since data are complex, their input terminals are 52 bits. This process is repeated for the third group of data, and at the same time firs four complex data are loaded into the latch group of NL. The OL latch outputs are connected together and are fed to the static RAM by a 54-bit bus. Therefore, OL latches must be tri-state like Fig.14.

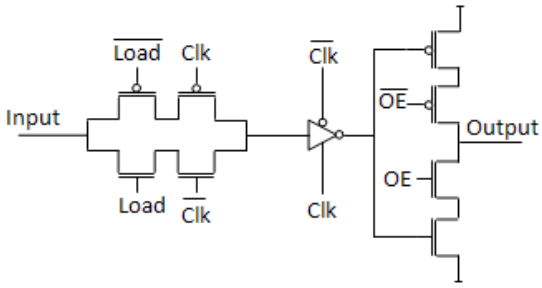


Fig.14. OL latch with load and output enable

We have used clocked NOT gate in implementation of latches. Fig.15. illustrates the details of clocked NOT gate.

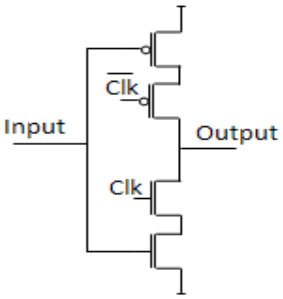


Fig.15. schematic of clocked NOT

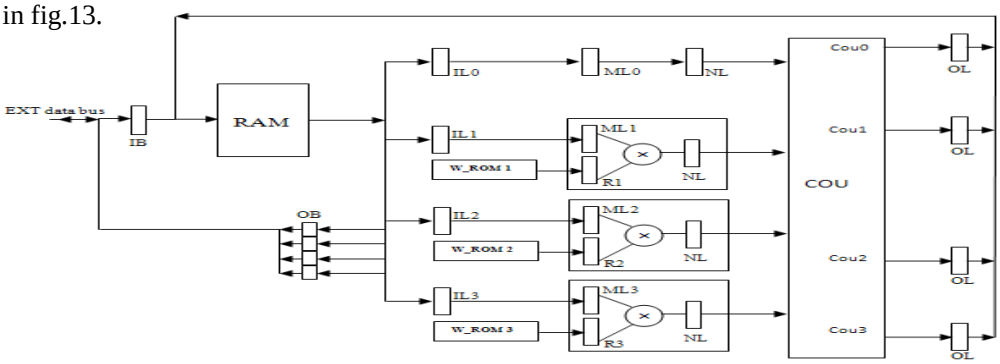


Fig.13. latches position in 4-point DFT processor

4 Conclusion

The need for high speed digital signal processing is rapidly increasing apparent with the current applications of synthetic aperture radar and image recognition among the most noticeable. The DFT is a powerful tool for measuring the spectral content of sampled signals.

A FFT processor dedicated to compute 64-point FFT has been designed using 0.35µm CMOS process. This processor computes 64-point FFT within 18.87µs using a 100-MHZ clock pulse. Results of designed processor are summarized in table1.

Transfer time	18.87 µs
Clock rate	100 MHZ
Chip area	13.3 mm^2
Power dissipation	645mw
Technology	0.35µm 2P4M CMOS

Table.1. performance parameters

References:

[1] J. G. Proakis and D. G. Manolakis, Digital Signal Processing principles, algorithms, and applications, 2nd ed, New York, Macmillan, 1992.

[2] M. Mano, Digital Design, New Jersey, Prentice –Hall, 1984

[3] F. Klass and M. J. Flynn, "A 16×16 bit static CMOS wave-pipelined multiplier"

[4] G. Langholz, J. Francioni, and A. Kandel, Elements of Computer Organization, New York, Wiley, 1990.

[5] M.K.Lee, K.W.Shin, and J.K.Lee, " A VLSI array processor for 16-point FFT," IEEE J.Solid- State Circuits, vol. 26, no. 9, pp 1286-1292, September 1991