

ASIC Implementation of High Speed Processor for Calculating Discrete Fourier Transformation using Circular Convolution Technique

P. Saha¹, A. Banerjee², A. Dandapat³, P. Bhattacharyya^{4*}

¹School of VLSI Technology, Bengal Engineering and Science University, Shibpur, Howrah-711103, WB, INDIA.

²Department of Electronics and Communication Engineering, JIS College of Engineering. Kalyani-741235, WB, INDIA.

³Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata-700032, WB, INDIA.

⁴Department of Electronics and Telecommunication Engineering, Bengal Engineering and Science University. Shibpur, Howrah-711103, WB, INDIA.

Email: sahaprabir1@gmail.com; banerjee.arindam1@gmail.com; anup.dandapat@gmail.com; pb_etc_besu@yahoo.com

*Corresponding author: Tel.: +913326684561; fax: +913326682916
Email: pb_etc_besu@yahoo.com

Abstract: - The improvement in speed and power for calculating discrete Fourier transformation using circular convolution is well established, but all the work so far been reported are at FPGA (gate) level. In this paper ASIC implementation of high speed processor for calculating Discrete Fourier Transformation (DFT) based on circular convolution architectures is reported for the first time. The IEEE-754 single precision format was considered for the representation of the twiddle factors. The improvement of the speed for floating point multiplication/addition was achieved by canonical sign digit implementation methodology, which reduced the stages of operation significantly. The functionality of these circuits was checked and performance parameters such as propagation delay, dynamic switching power consumptions were calculated by spice spectre using standard 90nm CMOS technology. The implementation methodology ensure substantial reduction of propagation delay in comparison with systolic array and memory based implementation, most commonly used architectures, reported so far, for DFT processors. The propagation delay of the resulting 16 point DFT processor is only 23.79 μ s while the power consumption of the same was 14.32mW only for a layout area of \sim 12mm². Almost 50% improvement in speed from earlier reported DFT processors, e.g. systolic array and memory based implementation methodology, has been achieved.

Key-Words: - DFT, FFT, Circular convolution, Multiply and accumulate (MAC), Canonical sign digit (CSD) adder, CSD Multiplier.

1 Introduction

Discrete Fourier Transformation (DFT) is of immense importance in the field of Digital Signal Processing (DSP), Digital Image Processing (DIP), data compressions, high speed broadband communication, general filter design and convolutions [1-4] etc. Almost all the design techniques require a large amount of precisions for the DFT computation [4]. Moreover, optimized circuit implementation in terms of low hardware usage, reduction of the propagation delay and power consumption is essential for many wireless

applications [5]. On account of the wide range of the applications it is inevitable for the researchers to implement the ASIC processor for high speed DFT computation techniques.

A substantial amount of work has so far been reported on DFT processor [2-18] for speed improvement and power reduction, such as systolic array, reduced memory size, distributed arithmetic and CORDIC based implementations etc. The greatest disadvantages of Systolic array based implementation [2-3, 6-7] are huge area consumption due to presence of multipliers, while

distributed arithmetic [8-10] and memory based implantation [14] suffers from large ROM size requirement to accommodate the continued product. To the contrary, bottleneck of CORDIC based implementation [11-13] is its large overhead of pre/post processing units. Recently Benhamid et. al. [17] reported on radix 2² Genetic Algorithm (GA) based Canonical Sign Digit (CSD) multiplier less architecture for DFT processor. But, all of these techniques suffer from the limitations owing to large pre/post processing elements, and/or a large ROM size.

At algorithmic and structural level lot of implementation techniques have already been reported [2,3,5-8, 13,14,17] but to the best of our knowledge till date there is no report on transistor level (ASIC) implementation of such DFT processors. Moreover, most of the works reported so far, deals with the theoretical aspects of DFT processors, and did not fully discuss the practical circuit design issues like speed, power & layout area. In this paper we report of ASIC implementation of high speed DFT processor. The proposed techniques have been implemented using the reformulation of the transformation equations into cyclic convolution formation. To implement the hardware architecture for circular convolution of two N point sequences, MAC based architecture has been proposed, which uses systolic array for generation of the convolution sums. The proposed architecture neither imposes any limits on the method for calculation of convolution sum; nor does introduce round off errors. CSD multiplication/addition methodologies have been considered, to reduce the propagation delay of such DFT processors. On account of the CSD algorithms, multiplication/addition can be performed in constant time which is independent of the number of bits [19].

The proposed DFT processor architecture is fully optimized for N point DFT computation. The functionality of the circuits was designed and verified by Spice Spectre in 90nm CMOS technology. Proposed algorithm ensures substantial reduction in propagation delay and power consumptions compared to systolic array [2], distributed arithmetic [8], reduced memory [14], and radix 2² based implementation methodologies. Propagation delay for the proposed 16 point DFT processor was only 23.79µs with only 14.32mw power consumptions for a layout area of ~12mm².

2 Algorithm formulation for Discrete Fourier Transformation (DFT)

The Discrete Fourier Transform (DFT) of discrete signal $x(n)$ can be directly computed as:

$$X(k) = \sum_{n=0}^{N-1} x(n)w_N^{nk} \quad k=0,1,\dots,N-1 \quad (1)$$

Where $w_N = e^{-j2\pi/N} = \cos \frac{2\pi}{N} - j\sin \frac{2\pi}{N}$ and w_N is called in phase or twiddle factor and " $j^2=-1$ ". Here $x(n)$ and $X(k)$ are the sequences of the complex numbers.

An efficient method of computing the DFT that significantly reduces the number of required arithmetic operations is called FFT [20-21]. An FFT algorithm divides the DFT calculation into many short-length DFTs and results in huge savings of computations. If the length of DFT $N=R^v$, i.e., the product of identical factors, the corresponding FFT algorithms are called Radix-R algorithms. Assume FFT length is 2M, where M is the number of stages. The radix-2 DIF FFT divides an N-point DFT into 2, N/2-point DFTs, then into 4, N/4-point DFTs, and so on. That is, the radix-2 DIF FFT expresses the DFT equation as two summations, and then divides it into two equations, each of which computes every two output samples. To arrive at a two-point DFT decomposition, considering $w_N^{2nr} = w_{N/2}^{nr}$ and the following equations are derived by

$$X(2k) = \sum_{n=0}^{N/2-1} [x(n) + x(n + \frac{N}{2})]w_{N/2}^{nk} \quad (2)$$

Where $k = 0,1, \dots, (N/2 - 1)$

$$X(2k + 1) = \sum_{n=0}^{N/2-1} [x(n) - x(n + \frac{N}{2})]w_N^n w_{N/2}^{nk}$$

Where $k = 0,1, \dots, (N/2 - 1)$

Above equations are frequently represented in butterfly format. The butterfly of a Radix-2 algorithm is shown in Fig. 1(a).

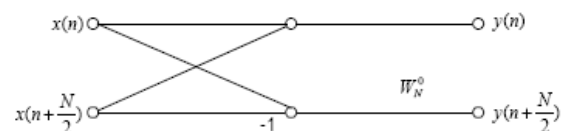


Fig. 1. (a)

The complete flow graph of an N-point Radix-2 FFT can be constructed by applying the basic butterfly structure (Fig.1. (a)) recursively, where N

= 2,4,8,... For an N-point Radix-2 FFT, it has $\log_2 N$ stages. Within stage s , for $s = 1, 2, \dots, \log_2 N$, there are $N/2^s$ groups of butterflies, with 2^{s-1} butterflies per group. The computation of the 8-point DFT, for instance, can be accomplished by the algorithm depicted in Fig. 1(b).

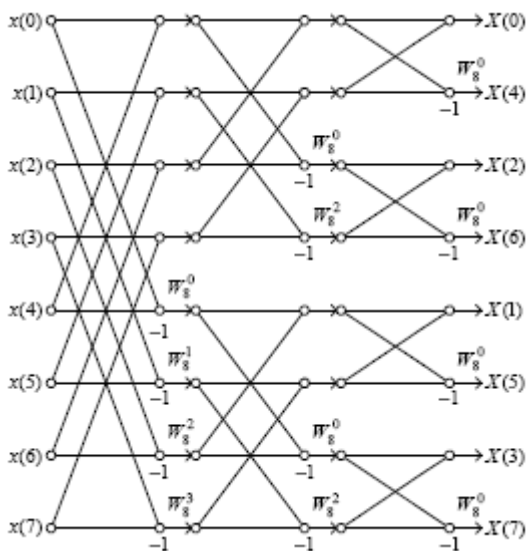


Fig. 1. (b)

Fig. 1. (a) Basic Butterfly structure of DFT
(b): Data flow graph of Butterfly structure for DFT

2.1 Cyclic Convolution formulation form Mathematical Expressions of DFT

For prime length DFT we can formulate equation (1) as

$$Y(0) = \sum_{n=0}^{N-1} x(n) \tag{4}$$

And

$$Y(g^k) = T(g^k) + x(0), \quad k = 1, 2, \dots, N-1 \tag{5}$$

Where

$$T(g^k) = \sum_{i=1}^{N-1} x(i) e^{-j \frac{2\pi}{N} g^{i+k}}, \quad k = 1, 2, \dots, N-1 \tag{6}$$

And “ g^j ” denotes the “ g^i modulo N ” operation. $T(k)$ is the cyclic convolution of the sequence $\{x(i), i=1, 2, \dots, N-1\}$ and the kernels $\{e^{-j \frac{2\pi}{N} g^{i+k}} \text{ where } i \text{ and } K = 1, 2, \dots, N-1\}$.

Considering 5 point DFT, as an example. The input sequence is given as $\{x(n), n=0, 1, 2, 3, 4\}$ and the Kernel is $(W = e^{-j \frac{2\pi}{5}})$, Then the equation (1) can be expressed as:

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & w^1 & w^2 & w^3 & w^4 \\ 1 & w^2 & w^4 & w^6 & w^8 \\ 1 & w^3 & w^6 & w^9 & w^{12} \\ 1 & w^4 & w^8 & w^{12} & w^{16} \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} \tag{7}$$

Alternatively equation (7) can be written as:

$$Y(0) = \sum_{n=0}^4 x(n)$$

And

$$\begin{bmatrix} Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \\ x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix} = \begin{bmatrix} x(1) & x(2) & x(4) & x(3) \\ x(3) & x(1) & x(2) & x(4) \\ x(4) & x(3) & x(1) & x(2) \\ x(2) & x(4) & x(3) & x(1) \end{bmatrix} \times \begin{bmatrix} W^2 \\ W^4 \\ W^3 \\ W^1 \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix} \tag{8}$$

The first part of RHS of equation (8) shown above is representing the 4 point circular convolution.

3 Hardware Implementation of DFT

In this section, we have kept our focus on designing simplified as well as efficient hardware architecture for the purpose of evaluating DFT algorithm. Simple trigonometric identities have been applied for the implementation of the various twiddle factors. The overall block diagram for the computation of the discrete Fourier Transformation processor is shown in Fig. 2.

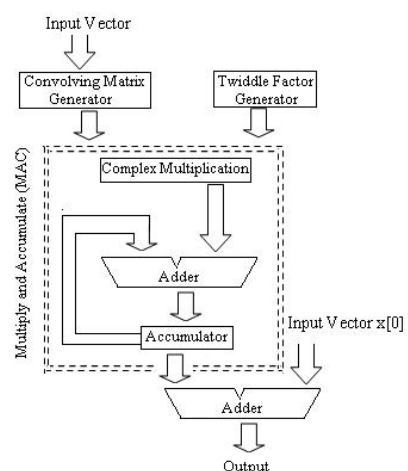


Fig. 2. Proposed Flow chart diagram for the ASIC of the high speed processor for calculating DFT

There are four blocks in the overall architecture. In the first block convolving matrices for the input data points is generated using Matrix Vector Rotation

methodology (MVR). At the same time, in the next block twiddle factors are generated by the twiddle factor generating circuitry. During the time of the multiplication operation, special types (Complex Multiplication) of the multiplication circuit has been implemented and incorporated with MAC. And finally at the last stage, CSD adder circuit has been used for the addition of the output of MAC, and fixed point input vectors.

has been employed for this purpose with a feedback connection from SS_3 output to SS_4 input via two block of serial to parallel converter and parallel to serial converter as shown in Fig. 3. Initially the line **MUX Enable** is set low in order to load the input vector X to the registers. After the input sequence is loaded into the registers, **MUX Enable** is set high to

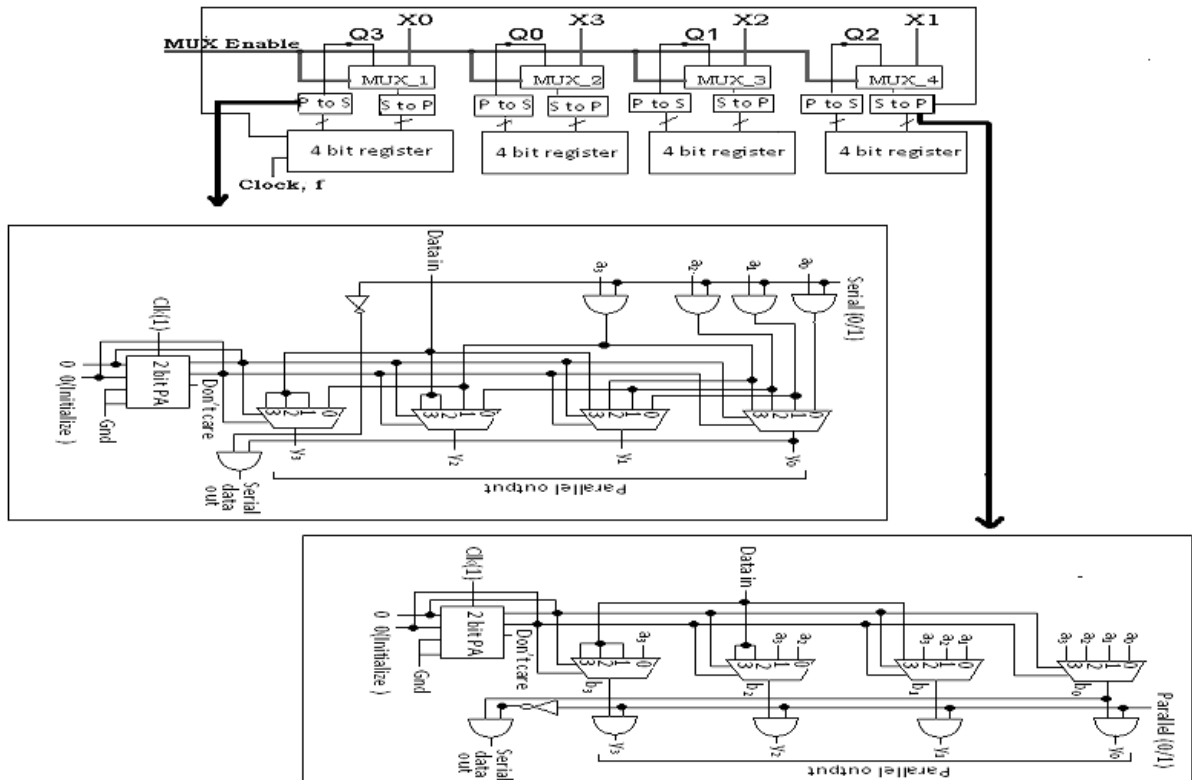


Fig. 3. 4×4 Convolution Matrix generation by Input Vector Rotation

3.1 Generation of the Convolution Matrix

As defined in equation the cyclic convolution between two 4-point sequences x and w can be expressed in matrix form as:

$$\begin{bmatrix} Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \end{bmatrix} = \begin{bmatrix} x(1) & x(2) & x(4) & x(3) \\ x(3) & x(1) & x(2) & x(4) \\ x(4) & x(3) & x(1) & x(2) \\ x(2) & x(4) & x(3) & x(1) \end{bmatrix} \times \begin{bmatrix} w^2 \\ w^4 \\ w^3 \\ w^1 \end{bmatrix} \quad (9)$$

It is to be noted that on the right hand side of equation (9), each row of the 4×4 square matrix can be generated by serial rotation of the input vector X = [x(1),x(2),x(4),x(3)] from left to right. A 4-bit right shift register {SS_4→SS_1→SS_2→SS_3}

perform the serial rotation on the arrival of each subsequent negative clock edge. The bus line **MUX Enable** is again set low after the generation of the fourth row of the square matrix before the arrival of the fourth clock edge.

The architecture shown in Fig. 3 has been dedicated for the generation of the elements of the column matrix [Y]. This hardware module imposes a serious restriction on the time period of the system clock. The time period of the system clock “f” should be large enough to allow the evaluation of each Y_i’s to be done within a single clock period. After generating the convolving matrix the convolving matrix is multiplied by the input sequences, and promoted to the input of the MAC.

3.1.1 Serial to Parallel Converter (S to P)

The functionality of the circuit can be achieved by clock triggered Serial in Parallel out shift registers and demultiplexers. The selection inputs to the demultiplexers are fed from an auto generated counter which is driven by a clock signal input. In this paper to avoid the sequential mechanism a fully combinational scheme for serial to parallel conversion has been adopted. The RTL representation of the combinational serial to parallel converter is shown in Fig. 3. This circuit is providing parallel output as well as serial output depending upon the selection input (Parallel). If 'Parallel' input is high then parallel outputs will be taken from the pins y_0 to y_3 which is indicated in Fig. 3. The clock driven shift registers has been replaced by parallel multiplexer shifters. The elements (bits) needed to be shifted are fed to the multiplexers in parallel as shown in Fig. 3. The select inputs to the multiplexers are fed from a parallel adder which is acting as a combinational counter. The clock input is replaced by a trigger input which is fed to the 'carry in' pin of the parallel adder. Again if any zero or one padding is needed then the bit input is fed to the Data in pin at the input side. At the output side the serial and parallel operation is monitored by AND gate arrays which is activated or deactivated depending upon the 'Parallel' input. This particular circuit is devised to execute right shift operation. That is why the serial output is taken from the LSB (b_0) bit. For left shifting the orientation of the inputs to the multiplexers will be reverse and the serial output will be taken from the MSB (b_3) bit.

3.1.2 Parallel to Serial Converter (P to S)

Parallel to Serial converter has the just reverse mechanism that of Serial to Parallel converter. The functionality of the circuit can be achieved by clock driven Parallel in Serial out shift registers and multiplexers. In this paper the clock triggered shift registers has been replaced by multiplexers connected in parallel and the selection inputs are monitored by a parallel adder which is acting as a combinational counter to avoid the clock triggering and clock skewing. The RTL representation of the combinational parallel to serial converter is shown in Fig. 3. The parallel or serial mechanism is obtained by a selection input (Serial). If the 'Serial' input is low the serial

data will be taken at the serial data out pin. The particular architecture is devised to achieve parallel outputs as well as serial output depending upon the selection input 'Serial'. The parallel inputs are fed to the AND gate array which is activated or deactivated by the 'Serial' input. The multiplexer inputs are connected in shifted fashion shown in Fig. 3. This orientation is maintained to execute the right shift operation. The same reason forces us to take the serial output from LSB (y_0) of the parallel output. For the left shift operation the orientation of the input bit array will be reverse. Similarly the serial output will be taken from the MSB (y_3) of the parallel output bit array in case of left shift operation. If any zero or one padding is needed the corresponding input is fed to the 'Data in' pin.

3.2 Canonical Sign Digit

The second area reduction technique that is used attempts to reduce the number of 1's required in a coefficient's power-of-two representation. Using a canonical signed digit (CSD) representation, coefficients can be represented using the fewest number of non-zero bits [24].

3.2.1 Canonical Sign Digit Adder

Carry propagation free CSD addition is performed in two steps.

I. Determining the intermediate carry $\{C_i \in (\bar{1}, 0, 1)\}$ and intermediate sum digits $\{S_i \in (\bar{1}, 0, 1)\}$, satisfying the condition $x_i + y_i = z_i + C_{i-1}$. Where x_{i+1} and y_{i+1} are the augends and addend digits respectively.

II. Obtain the sum digits $\{Z_i \in (\bar{1}, 0, 1)\}$ at each position by adding the intermediate sum digits S_i and C_i from the next lower order positions.

3.3.2 Canonical Sign Digit Multiplier

In general N bit floating point parallel multiplication, $N \times N$ partial products are generated first and then added to obtain the product [27]. The partial products may be added by using Full adder or Full adder and compressors. In our algorithm, we add partial products pair-wise by means of CSD adders. We represent all intermediate results in CSD

format and perform all additions using CSD adders. Finally, we convert the product into binary representation.

Multiplication Algorithm

<Input>

X and Y : Multiplicand and multiplier respectively (Both are N Bits). Both are signed digit floating point numbers.

<Output>

Sum : the products of X and Y

Algorithm

Step 1: Generate N×N bits partial products using Baugh-Wooley's method.

Step 2: Add the partial products using CSD adders. Perform the additions at each level in the tree in parallel.

Fig. 4(a) illustrates an example of 4 bit 2's complement signed numbers multiplication. From the CSD multiplication algorithm it can be observed that, multiplication algorithm consists two parts.

- (I) Partial Product Generation.
- (II) Partial Product Addition.

Partial product generation is described in this section and the partial product addition stage is already described in the previous section.

CSDC Based Partial Product Generator:

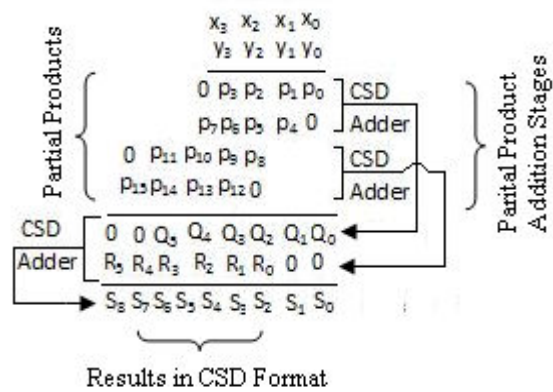
In normal array multiplication the partial products can be achieved by normal AND operation. In the CSD architecture the partial products have been generated by CSDC encoding technique, because CSD encoding techniques consider positive as well as negative sign. In this technique the AND operation has been accomplished by considering each bit including its sign. Partial product generation technique for CSD multiplication can be implemented from Table 1, and the gate level implementation for Table 1 is shown in Fig. 4 (b). The Boolean expressions for the partial product P_i and its sign ($signP_i$) are expressed below.

$$P_i = x_i y_i \tag{10}$$

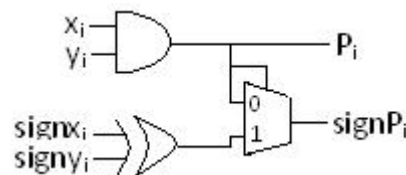
$$signP_i = P_i (Signx_i \oplus Signy_i) \tag{11}$$

Table 1 Truth Table implementation for partial Product generation

x_i	y_i	P_i
0	0	0
0	1	0
1	0	0
1	1	1
0	$\bar{1}$	0
$\bar{1}$	0	0
1	$\bar{1}$	$\bar{1}$
$\bar{1}$	1	$\bar{1}$
$\bar{1}$	$\bar{1}$	1



(a)



(b)

Fig. 4. (a) CSD Multiplication Technique; (b): Gate Level implementation Diagram for partial products

3.3 Twiddle Factor Generation using Minimum Constant Multiplication

In this subsection we are concentrating on the generation of real and imaginary parts of different twiddle factors. The resolution of this generation is based on the factor resolution around the unit circle of w_N . For a small range of the twiddle factor, algebraic method is advantageous to calculate the twiddle factors, because only four terms i.e., (+1,-

1,+j,-j) are generated for that case. For large values of 'n' we are simple considering the octave

a W_{16} multiplier using only two multipliers with the constant values $\sin\frac{\pi}{8}$ and $\cos\frac{\pi}{4}$. Note that multiplication by two is equivalent to a left-shift,

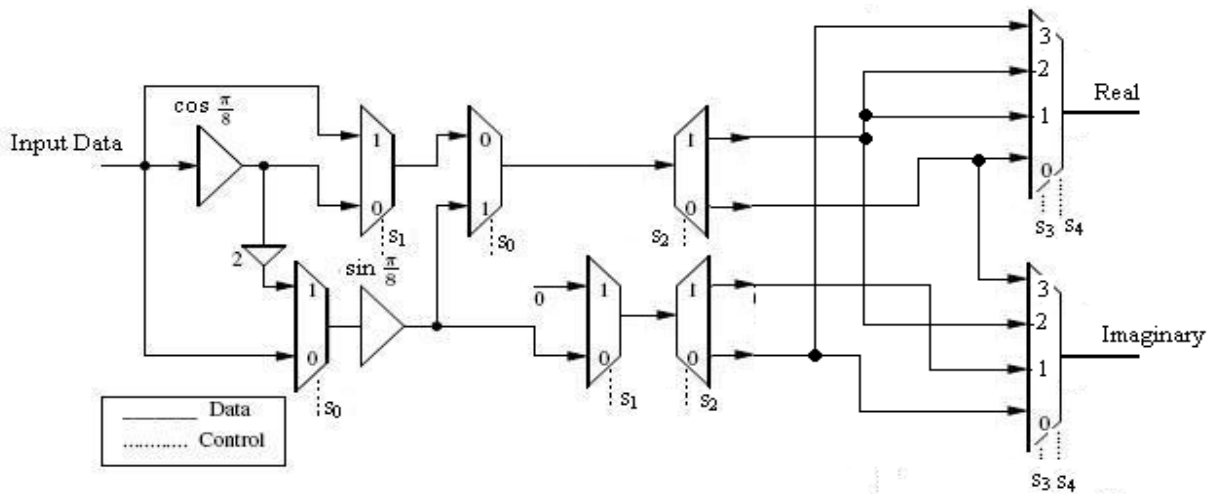


Fig. 5. Twiddle factor generation circuit

Table 2 Requirement of the control signal for twiddle factor generation

s_0	s_1	s_2	s_3	s_4	Real Part of twiddle factor	Imaginary Part of twiddle factor
0	1	0	0	0	1	0
0	0	1	0	1	$\cos\frac{\pi}{8}$	$\sin\frac{\pi}{8}$
1	0	1	1	0	$\cos\frac{\pi}{4}$	$\sin\frac{\pi}{4}$
0	0	1	1	1	$\sin\frac{\pi}{8}$	$\cos\frac{\pi}{8}$
0	1	0	1	1	0	1

symmetry of the twiddle factors, because all the the range of the twiddle factor lies in between the range of $(0 < \beta < \pi/4)$. For a W_8 -multiplier this leads to that only a multiplication by either 1 or $\sin\frac{\pi}{4}$ ($=\cos\frac{\pi}{4}$) is required. This can easily be realized using a multiplexer selecting between the input and the output of a constant multiplier with coefficient $\sin\frac{\pi}{4}$. For the W_{16} multiplier a number of different approaches have been proposed. In [26] a W_{16} -multiplier based on the trigonometric identity $\sin 2\beta = 2 \sin\beta\cos\beta$ was introduced. Hence, as $2\frac{\pi}{8} = \frac{\pi}{4}$ is possible to compute all the three required values for

and, hence, is not considered as a multiplication. The structure shown in Fig. 5 is slightly modified compared to that in [26] as two multiplexers and two de-multiplexers are added at the output to allow multiplication by 1 in the structure. The control signal requirement for generation of the twiddle factors are summarized in table 2.

3.4 Multiplier and Accumulator (MAC)

MAC is the composition of adders, complex multipliers and accumulators. Complex multiplier and adder delays play an important role for the design of MAC. For the enrichment of the speed operation, Canonical Signed Digit [19] is used. One

implementation of the multiplier could be as a canonical signed digit multiplier [19]. The inputs of MAC are coming from two external circuitry, i.e., twiddle factor generation circuits and input vector rotational circuits. The multiplication circuits are performing the complex multiplication and give the results to the adder. The function of the adder block is performing the accumulation of the results, and then the results are stored in the memory locations. The function of the conventional MAC unit is given by the following equation:

$$F = \sum_i a_i b_i \tag{12}$$

Fig. 6 indicates the functional block diagram of the MAC. The design consists of one N CSD multiplier [19], one N+2 bit accumulator register, one control logic/DeMUX block, one N bit register. The two N bit numbers are multiplied and stored in 2Nbit registers. In first clock pulse the numbers are multiplied and the result is added with zero.

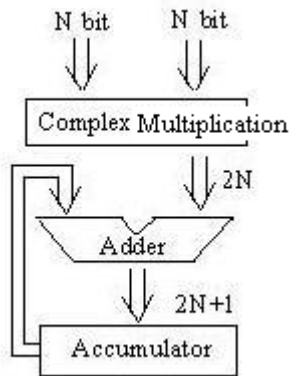


Fig. 6. Functional Block diagram of MAC

4 Results and Discussion

Transistor level simulation was performed using Spice Spectre simulator using 90nm CMOS technology with 1 volt power supply. Dual threshold voltage (V_T) operating mode was considered for simulation to determine the performance parameters. The proper choice of threshold voltages for a particular transistor in the circuit is based on a number of logics as described below:

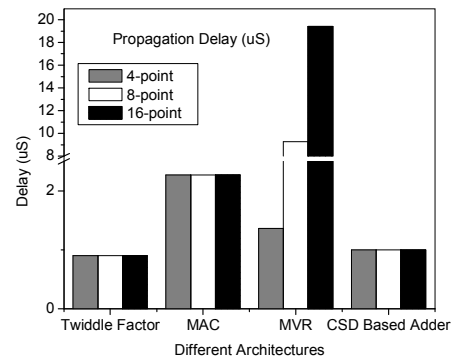
- a. Placement of high- V_T transistors on the leakage path directly between supply and

ground reduces the sub-threshold leakage current and hence static power.

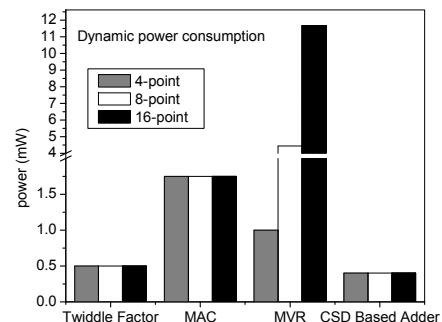
- b. Placement of low- V_T transistors on the signal propagation path from the input node to the output improves the performance substantially.
- c. A logical intersection of the conditions illustrated in (a) and (b) requires an optimized choice that leads to the minimum EDP.

Proper modifications at the device, circuit and architectural levels of design hierarchy reduce the Energy Delay Product (EDP) for the proposed design. Transmission Gates (TG) are used for the design of different modules for faster operation and better logic transformation.

Input data was taken in a regular fashion for experimental purpose. The delay and the power measured using the worst-case pattern and from the output where the delay is maximum. The individual performance parameters such as propagation delay, dynamic switching power consumption of the individual circuit modules, i.e., twiddle factor generator, MAC, CSD Adder, Input vector rotation matrix is shown in Fig. 7.



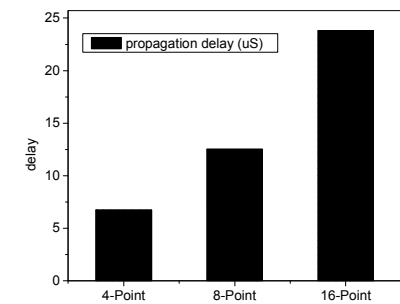
(a)



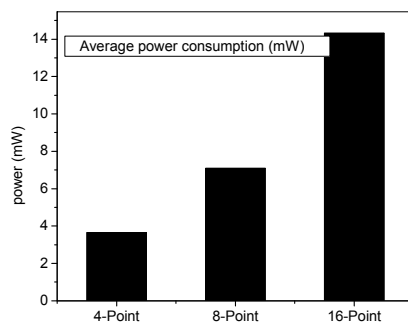
(b)

Fig. 7. (a) Propagation Delay (μs) (b) Dynamic average power consumption analysis of different circuit modules as a function of Input Number of Points.

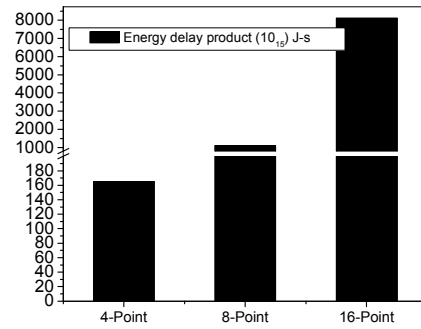
We focused our main concentration for reducing the propagation delay, dynamic average power consumption and energy delay product. Fig. 8 indicates the performance parameters such as propagation delay, and dynamic switching power consumptions and energy delay product analysis proposed DFT processor. All the mentioned designed have been simulated in using same technology through spice spectre simulator for the comparison purpose. Fig. 9 represents the graphical analysis of comparison results for performance parameters such as propagation delay and dynamic switching power consumption, and energy delay product of different architectures. From the simulation result analysis we can claim that, incorporation of TG with dual threshold voltage CMOS technology may be the plausible choice in future technology for high speed DFT processors.



(a)

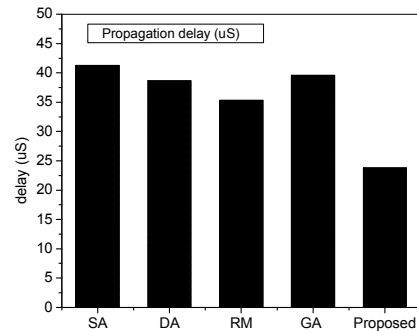


(b)

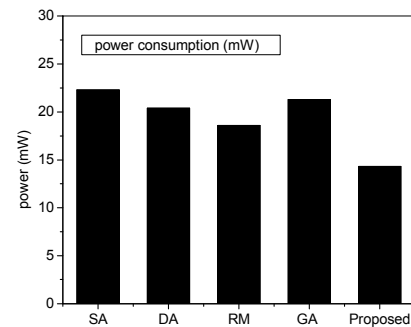


(c)

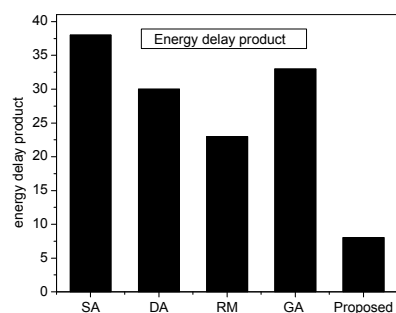
Fig. 8. (a) Propagation Delay (μs), (b) Dynamic Average Switching Power (mW), (c) Energy delay product (10^{-15} J-s) analysis of the proposed DFT processor as a function of Input Number of Points.



(a)



(b)



(c)

Fig. 9. Comparison results such as (a) Propagation Delay (μ s), (b) Dynamic Average Switching Power (mW) consumption (c) Energy delay product (10^{-12} J-s) analysis for different architectures.

Layout of the proposed 16 point DFT processor has been implemented using L-Edit V-13 of T-Spice simulator with 90 nm CMOS technology and is shown in Fig. 10. Layout area was found to be only $\sim 12 \text{ mm}^2$. The proposed DFT offered 73% and 55% improvement in terms of speed and power consumption respectively, in comparison with systolic array based implementation with operating voltage of 1v. Whereas, the corresponding improvement in terms of propagation delay and power was found to be 48% and 30% respectively, with reference to the reduced memory based implementation.

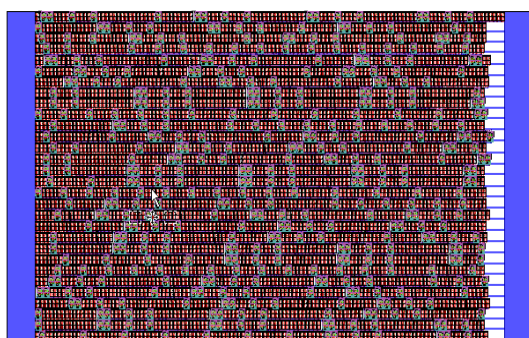


Fig. 10. Layout of 16-point Discrete Fourier Transformation (DFT) processor using Circular Convolution Technique. Layout was implemented using L-Edit V-13 of T-Spice simulator and area was $\sim 12 \text{ mm}^2$.

5 CONCLUSIONS

In this paper we report on transistor level implementation of a high speed DFT processor based on circular convolution technique. The implementation methodology of circular convolution architecture has been designed using MAC, which ensure the single kernel implementation, leading to the substantial reduction in the propagation delay. CSD multiplication/addition methodologies have been utilized, to increase the operating speed of such DFT processors. The transistor level implementation was carried out using Spice Spectre and obtained results were compared with the mostly used architectures like systolic array, distributed arithmetic, and reduced memory based implementation. The proposed DFT processor offered 73% and 55% improvement in speed and power consumption respectively, compared to systolic array based implementation. Whereas, the corresponding improvement in terms of propagation delay and power was found to be 48% and 30% respectively, with reference to the reduced memory based implementation.

References:

- [1] Y. Jiang, J. Peng, Discrete Fourier Transformations with Weight, in *Proceedings of the IEEE, International Conference on Information and Computing*, 2011, pp. 168-171
- [2] L.W. Chang, M.Y. Chen, A new systolic array for Discrete Fourier Transform, *IEEE Transaction on Acoustic Speech and Signal Processing*, Vol. 36, No. 10, 1988, pp. 1665–1666.
- [3] W.H. Fang, M.L. Wu, An efficient unified systolic architecture for the computation of discrete trigonometric transforms, in *Proceedings of the IEEE, International Symposium on Circuits and Systems*, 1997, pp. 2092–2095.
- [4] R. Sarmiento, F. Tobajas, V. Armas, R. E. Chain, J. F. Lopez, J. A. Montiel-Nelson, A. Nunez, A CORDIC Processor for FFT Computation and Its Implementation Using Gallium Arsenide Technology, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 1, 1998, pp. 18-30.
- [5] H. Ho, V. Szwarc, and T. Kwasniewski, Design and Implementation of a Multiplierless Reconfigurable DFT/DCT Processor, in *Proceedings of the IEEE, North-East workshop on Circuits and Systems and TAISA conference*, 2009, pp. 1-4.

- [6] J.A. Beraldin, T. Aboulnasr, W. Steenart, Efficient one-dimensional systolic array realization of discrete Fourier transform, *IEEE Transactions on Circuit and Systems*, Vol. 36, No. 1, 1989, pp. 95–100.
- [7] Y. Jiang, T. Zhou, Y. Tang, Y. Wang, Twiddle-Factor-Based FFT Algorithm with Reduced Memory Access, in *Proceedings of the IEEE, Parallel and Distributed Processing Symposium*, 2002, pp. 70-77.
- [8] H.C. Chen, J.I. Guo, C.W. Jen, T.S. Chang, Distributed arithmetic realization of cyclic convolution and its DFT application, *IEE proceedings on Circuits, Devices and Systems*, Vol. 152, No. 6, 2005, pp. 615-629.
- [9] W.C. Siu, C.F. Chen, New realisation technique of high-speed discrete Fourier transform described by distributed arithmetic, *IEE proceedings on Computer and Digital Techniques*, Vol. 130, No. 6, 1983, pp. 177-182.
- [10] C. Cheng, K.K. Parhi, Low-Cost Fast VLSI Algorithm for Discrete Fourier Transform, *IEEE Transaction on Circuits and Systems-I*, Vol. 54, No. 4, 2007, pp. 791-806.
- [11] Jayashankar, Efficient computation of the DFT of a 2N Point Real Sequence using FFT with CORDIC based Butterflies, in *Proceedings of the IEEE, TENCON*, 2008, pp. 1-5.
- [12] P.S. Kumar, K.M.M. Prabhu, Novel CORDIC-based systolic arrays for the DFT and the DHT, in *Proceedings of the IEEE, High Performance Computing on the Information Superhighway*, 1997, pp. 547-551.
- [13] T-Y. Sung, H-C. Hsin, L-T. KO, Reconfigurable VLSI Architecture for FFT Processor, in *WSEAS Transactions on Circuits and Systems*, Issue 6, Vol. 9, June 2009, pp. 465-474
- [14] J.I. Guo, C.M. Liu, C.W. Jen, The efficient memory-based VLSI array designs for DFT and DCT, *IEEE Transactions on Circuits Systems- II, Analog and Digital Signal Processing*, Vol. 39, No. 10, 1992, pp. 723–733.
- [15] Y-S. Shieh, T-Y. Sung, H-C. Hsin, A Novel Linear Array for Discrete Cosine Transform, in *WSEAS Transactions on Circuits and Systems*, Issue 5, Vol. 9, May 2010, pp. 335-346
- [16] J.I. Guo, An efficient parallel adder based design for one dimensional discrete Cosine transform, *IEE proceedings on Circuits, Devices and Systems*, Vol. 147, No. 5, 2000, pp. 276-282.
- [17] M. Benhamid, M.B. Othman, Hardware Implementation of a Genetic Algorithm Based Canonical Signed Digit Multiplier-less Fast Fourier Transform Processor for Multiband Orthogonal Frequency Division Multiplexing Ultra Wideband Applications, *Journal of Mathematics and Statistics*, Vol. 5, No.4, 2009, pp. 241-250.
- [18] P.A. Milder, F. Franchetti, J. C. Hoe, M. Puschel, Hardware Implementation of The Discrete Fourier Transformation With Non-Power-of-Two Problem Size, in *Proceedings of the IEEE, International conference on Acoustics Speech and Signal Processing*, 2010, pp. 1546-1549.
- [19] P. Saha, A. Banerjee, I. Banerjee, A. Dandapat, High Speed Low Power Floating Point Multiplier Design Based on CSD (Canonical Sign Digit), *IEEE symposium on VLSI Design and Testing, VDAT-10*, 2010.
- [20] G.D. Bergland, A Radix-Eight Fast-Fourier Transform Subroutine for Real-Valued Series, *IEEE Trans. Audio Electroacoustics*, Vol. 17, No. 2, 1969, pp. 138-144.
- [21] C.S. Burrus, T.W.Parks, *DFT/FFT and Convolution Algorithms and Implementation*, NY: John Wiley & Sons, 1985.
- [22] J.W. Pierre, A Novel Method For Calculating The Convolution Sum of Two Finite Length Sequences, *IEEE Transactions on Education*, Vol. 39, No. 1, 1996, pp. 77-80.
- [23] H.T. Kung, Why systolic architectures? *Computer Mag.*, Vol. 15, 1982, pp. 37–45.
- [24] A. Avizienis, Signed-digit number representations for fast parallel arithmetic, *IRE Trans. Electron. Computer*, Vol. EC-10, 1961, pp. 389-400.
- [25] IEEE Standard 754 for Binary Floating-Point Arithmetic, 1996.
- [26] J.E. Oh, M.S. Lim, New radix-2 to the 4th power pipeline FFT processor, *IEICE Trans. Electron*, Vol. E88-C, No. 8, 2005, pp. 1740–1764.
- [27] C.-H. Lin, S.-C. Yi, J.-J. Chen, A Multiplier Based on the Algorithm of Chinese Abacus, in *WSEAS Transactions on Electronics*, Issue 1, Vol. 6, Jan 2009, pp. 11-22.