A High Throughput FFT Processor With No Multipliers

Shakeel S. Abdulla[†], Haewoon Nam[*], Mark McDermot[†], and Jacob A. Abraham[†]

[†]Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712, USA
[*]Mobile Devices Technology Office, Motorola Inc, Austin, TX 78730, USA
Email: shakeel.abdullah@gmail.com, haewoon_nam@ieee.org, mcdermot@ece.utexas.edu, jaa@cerc.utexas.edu

Abstract— **A novel technique for implementing very high speed FFTs based on unrolled CORDIC structures is proposed in this paper. There has been a lot of research in the area of FFT algorithm implementation; most of the research is focused on reduction of the computational complexity by selection and efficient decomposition of the FFT algorithm. However there has not been much research on using the CORDIC structures for FFT implementations, especially for large, high speed and high throughput FFT transforms, due to the recursive nature of the CORDIC algorithms. The key ideas in this paper are replacing the sine and cosine twiddle factors in the conventional FFT architecture by non-iterative CORDIC micro-rotations which allow substantial ($\sim 50\%$) reduction in read-only memory (ROM) table size, and total removal of complex multipliers. A new method to derive the optimal unrolling/unfolding factor for a desired FFT application based on the MSE (mean square error) is also proposed in this paper. Implemented on a Virtex-4 FPGA, the CORDIC based FFT runs 3.9 times faster and occupies 37% less area than an equivalent complex multiplier-based FFT implementation.**

I. Introduction

The Fast Fourier Transform (FFT) algorithm is a computationally efficient way to implement the Discrete Fourier Transform. This is a fundamental operation in the increasingly popular multicarrier modulation technologies, such as orthogonal frequency division multiplexing (OFDM) and discrete multitone (DMT), in current and emerging high-speed data communication systems. The implementation of an FFT processor is one of the most challenging parts in order to meet real-time processing requirements in such systems while achieving reduced complexity [1].

The FFT hardware is heavily constrained by the power and area requirements [2], [3], and thus the focus is to minimize these two parameters without sacrificing the performance. In line with this, designers tend to use pipelined high speed multipliers, but these, in fact, add large area overhead. With sub-micron processes, the leakage power is a large percentage of the total power consumption. In order to reduce the leakage, the designer would have to ensure the FFT hardware occupies less area (i.e., using fewer transistors). For large FFT sizes the major portion of the area for the FFT hardware generally comes from the storage/memory elements for the twiddle factor tables and pipeline registers. The computational units such as complex multipliers and the complex adders also contribute to the area of the FFT significantly. However, the contribution of the control logic for the FFT algorithms on the area is relatively very small.

Several architectures have been proposed for FFT implementations. FFT architectures can be classified broadly based on the decimation algorithms used, i.e., decimation in time (DIT) or decimation in frequency (DIF). The DIF class of algorithms has been found to introduce less computational noise for fixed point hardware implementations. [4] compares the computational noise for fixed point DIF and DIT implementations.

The most well-known DIF architectures are the delay commutation, delay feedback [5], [6] and in-place computation [1], [7]–[9]. Our main focus is on the delay feedback (DF) architecture since it is the most efficient one among the three architectures from the perspective of 100% pipeline utilization; additionally, it has a simple controller. The DF architecture can sustain a certain throughput and has finite latency. One of the major advantages for the DF architecture lies in its scalability, where a different FFT size can be readily supported due to the multi-stage structure. Therefore, the DF architecture is attractive for the fourth generation wireless communication systems, such as Worldwide Interoperability for Microwave Access (WiMAX) and 3rd Generation Partnership Project (3GPP) Long-Term Evolution (LTE), where a scalable system bandwidth is required.

Among these hardware-efficient algorithms is an FFT using Co-Ordinate Rotation DIgital Computer (CORDIC) algorithm, which employs an iterative approach with only shift and add operations to reduce the complexity of computations that are of transcendental nature [10]. It was the seminal work of Despain [11]–[13] that incorporated this technology for the first time in the FFT hardware design, which provides extremely low complexity using CORDIC structures but only works for modest or low data rates because of the recursive structure. Although there are several papers in the literature on standalone CORDIC implementations [13], [14], such as the radix-4 CORDIC [15] and the recoded CORDIC architecture [16], little is known in the literature about a standard method for designing and incorporating high speed and high throughput FFTs using a non-recursive CORDIC structure. Mondwurf [17] describes an FFT implementation using an unrolled CORDIC structure, where the angle values are stored in direct form. These angle values are then used to calculate the micro-rotation vector. This approach consumes larger area than the proposed implementation since it requires ROM tables, adders, and comparators for the computation of the micro-rotation vectors.

This paper highlights the design of a non-recursive CORDIC-based FFT implementation, where multipliers are eliminated and therefore computational complexity is purely defined by additions. Elimination of the twiddle factor ROM tables and introduction of the micro-rotation vector ROM is another contribution of this paper. Further, this paper describes a way of finding the optimum number of CORDIC micro-rotation stages, defined by the unrolling factor, based on mean square error (MSE) for a given FFT application.

II. FFT Structures and CORDIC

A. FFT Architectures

Fig. 1 shows a typical DF architecture for radix-2 (R2) 8-point FFT system whose mode of operation is given as follows. The 1st-stage uses 4-delay pipeline registers to wait for the 5th sample to arrive before it is combined with the 1st sample that is held in the feedback pipeline registers. The difference is multiplied by the twiddle factors and is passed to the 2nd-stage, while the sum is circulated back in to the delay line which was holding the initial four samples of the data. After the four difference samples are passed to the 2nd-stage the four samples of the sum is passed to the 2nd-stage. The sum is not multiplied by twiddle factors. The 2nd and further stages work in a similar fashion except the 2nd stage uses two delay elements and the last uses one delay element. It is assumed that one sample of data is provided every clock cycle.

Note that the delay commutation has 50% pipeline utilization, while the in-place computation has more than 100% buffer/memory utilization in comparison with the DF architecture. However the in-place computation consumes more active power since it has to run faster to maintain the throughput. The in-place computation algorithm requires much less area [8], [18] in comparison with the DF architecture. Therefore, the leakage power dissipation is quite low for this type of architecture. In-place computation is very prominently used in many of the DSP processors.

B. CORDIC Structure

Transforming complex twiddle factor multiplications into CORDIC operations can eliminate the complex multiplications, whereas taking any generic decomposition of the FFT would still need complex multiplications. Therefore it can be deduced that any complex multiplier based FFT architecture has its CORDIC based equivalent, which may provide a simpler implementation. In general, complex multiplications of the form given as

$$X + jY = (x + jy)\, e^{-j\theta}, \tag{1}$$

can be represented in matrix form as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2}$$

Notice that a complex multiplication is typically equivalent to four real-valued multiplications and two real-valued additions. The rotation angle θ can be approximated with a linear combination of micro-rotations each of which has a different angle. Let $\mathbb{A} = \{a_0, a_1, \cdots, a_u\}$ denote the micro-rotation direction vector, where $a_i \in \{1, -1\}$, and $\boldsymbol{\Theta} = \{\arctan(1), \arctan(1/2), \arctan(1/4), \cdots\}$ be the set of micro-rotation angles. Consider that the length of $\mathbb{A}$ is (u+1), which is defined as the unrolling factor. Based on this approximation, the rotation matrix in (2) can be factorized into a product of micro-rotation matrices, where $\theta \approx \sum_{i=0}^{u} a_i\theta_i$, as

$$\begin{aligned} \begin{bmatrix} X \\ Y \end{bmatrix} &= \begin{bmatrix} \cos(a_0\theta_0) & \sin(a_0\theta_0) \\ -\sin(a_0\theta_0) & \cos(a_0\theta_0) \end{bmatrix} \times \cdots \\ &\times \begin{bmatrix} \cos(a_u\theta_u) & \sin(a_u\theta_u) \\ -\sin(a_u\theta_u) & \cos(a_u\theta_u) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \end{aligned} \tag{3}$$

which can be re-written as

$$\begin{aligned} \begin{bmatrix} X \\ Y \end{bmatrix} &= f_0 \begin{bmatrix} 1 & a_0(1/2^0) \\ -a_0(1/2^0) & 1 \end{bmatrix} \times \cdots \\ &\times f_u \begin{bmatrix} 1 & a_u(1/2^u) \\ -a_u(1/2^u) & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \end{aligned} \tag{4}$$

where $f_i = \cos(\theta_i)$ and $\theta_i = \tan^{-1}\left(\frac{1}{2^i}\right)$. This is further simplified as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \left(\prod_{i=0}^{u} f_i\right) \underbrace{\left(\prod_{i=0}^{u} \begin{bmatrix} 1 & a_i t_i \\ -a_i t_i & 1 \end{bmatrix}\right)}_{\text{Micro-rotation matrices}} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{5}$$

where $t_i = 1/2^i$, $\prod_{i=0}^{u} f_i$ is the scaling factor, and $f_i = \frac{1}{\sqrt{1 + \tan^2(\theta_i)}}$.

From the above mathematical derivations, it is easy to see that (1) can be approximated to (5). Note that the scaling factor is a constant value given a finite number of rotations. It can also be easily observed that this CORDIC-based structure does not need any twiddle factor memories, but needs to keep the micro-rotation direction vector $\mathbb{A} = \{a_0, a_1, ...a_i, ...a_u\}$ where each micro-rotation direction bit a_i determines the corresponding angle to be rotated either in the counter clock-wise or clock-wise direction.

III. Proposed CORDIC-Based FFT

A. Hardware Optimization for CORDIC Structure

As described in the previous section, the CORDIC structure is, in general, recursive where the micro-rotation matrices are multiplied for the computation of the rotation. In order to speed up the hardware and to increase the parallelism of the computations, the proposed FFT structure uses a hardware optimization technique based on a single product matrix, where the micro-rotation direction vector $\mathbb{A}$ for each angle is pre-calculated. Let T_{adder} be delay from the adder circuit, and T_{scale} be the delay from the scaler circuit. Then the direct

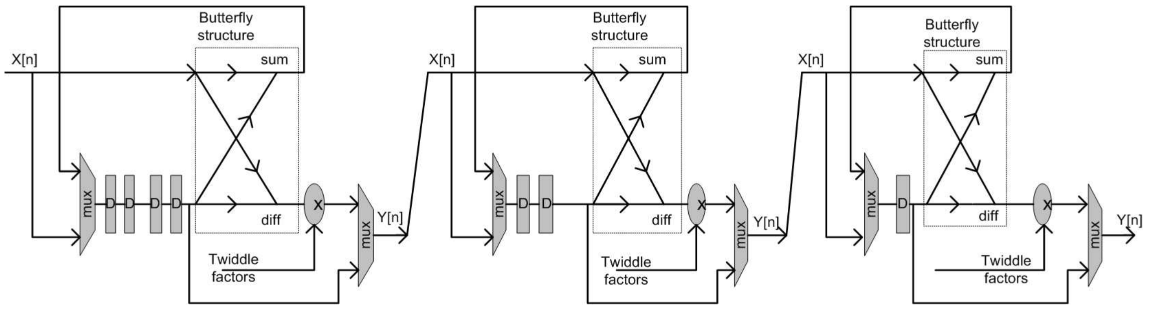
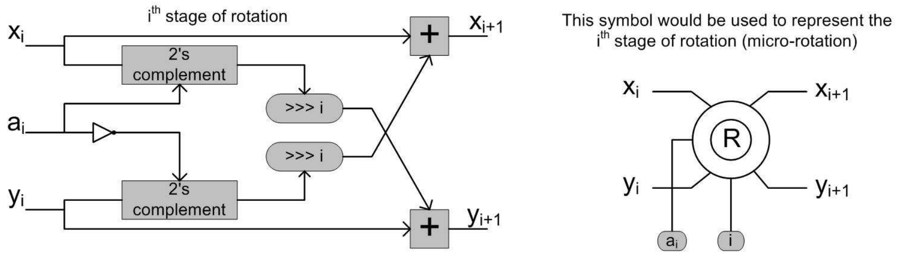
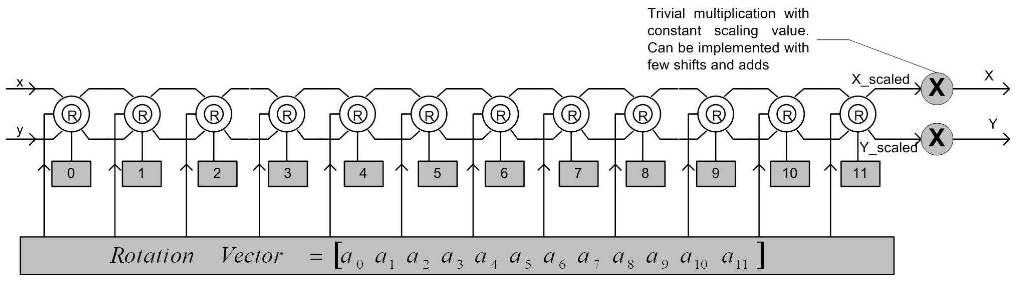


Fig. 1. Hardware Structure of Delay Feedback Architecture for Radix-2, 8-Point FFT.


ith stage of rotation
x_i
x_{i+1}
2's complement
>>> i
a_i
y_i
y_{i+1}
This symbol would be used to represent the ith stage of rotation (micro-rotation)
R
i

(a)


Trivial multiplication with constant scaling value. Can be implemented with few shifts and adds
x
y
X_scaled
Y_scaled
X
Y
0 1 2 3 4 5 6 7 8 9 10 11
Rotation Vector = [a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11}]

(b)

Fig. 2. Block diagram for (a) ith micro-rotation stage and (b) 12-stage unrolled CORDIC.

implementation of (5) leads to a slow hardware due to the critical path $T_{critical} = U * T_{adder} + T_{scale}$, where T_{scale} is easy to optimize for speed since it is a trivial multiplication. However, the optimization of $U * T_{adder}$ for speed requires the micro-rotation matrices to be combined so that the resultant matrix can be computed in parallel. This parallel approach may add more logic but it makes the critical path faster. This optimization can be found from the following derivations, where a single 2×2 composite matrix that represents the complete rotation by the vector $\mathbb{A}$ can be determined depending upon the number of micro-rotation stages, which is also called the unrolling factor. But identifying the optimal unrolling factor is contingent upon the acceptable quantization error for the application. For a given unrolling factor, the scaling factor becomes constant. Consider the unrolling factor of $(u + 1)$, (5) can be re-written as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \left(\prod_{i=0}^{u} f_i \right) \begin{bmatrix} U_0(\mathbb{A}) & U_1(\mathbb{A}) \\ -U_1(\mathbb{A}) & U_0(\mathbb{A}) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{6}$$

where

$$\begin{aligned} U_0(\mathbb{A}) &= 1 + \sum_{k=1}^{\lfloor (u+1)/2 \rfloor} (-1)^k \sum_{i \in \mathcal{S}_{2k}} \alpha_i \tau_i \\ U_1(\mathbb{A}) &= \sum_{k=1}^{\lceil (u+1)/2 \rceil} (-1)^{k-1} \sum_{i \in \mathcal{S}_{2k-1}} \alpha_i \tau_i. \end{aligned} \tag{7}$$

$\mathcal{S}_i$ is the set in which each element consists of i digits, each of which is equal to or less than u, and the set has all the elements for the possible combination using i digits. This structure can unroll the iterative CORDIC computation and allow a single matrix to be calculated given that the micro-rotation direction vector $\mathbb{A}$ is known in advance.

1) Example: Assuming the unrolling factor of 4, where $u = 3$, (7) can be written as

$$\begin{aligned} U_0(\mathbb{A}) &= 1 - \sum_{i \in \mathcal{S}_2} \alpha_i \tau_i + \sum_{i \in \mathcal{S}_4} \alpha_i \tau_i \\ U_1(\mathbb{A}) &= \sum_{i \in \mathcal{S}_1} \alpha_i \tau_i - \sum_{i \in \mathcal{S}_3} \alpha_i \tau_i, \end{aligned} \tag{8}$$

where $\mathcal{S}_1 = \{0,1,2,3\}, \mathcal{S}_2 = \{01,02,03,12,13,23\}, \mathcal{S}_3 = \{012,013,023,123\}$, and $\mathcal{S}_4 = \{0123\}$. Interestingly, $U_0(\mathbb{A})$ has the sets with even i values and $U_1(\mathbb{A})$ has the the sets with odd i values. Thus, (8) can be re-written as

$$\begin{aligned} U_0(\mathbb{A}) =& 1 - \alpha_{01}t_1 - \alpha_{02}t_2 - \alpha_{03}t_3 - \alpha_{12}t_3 \\ & - \alpha_{13}t_4 - \alpha_{23}t_5 + \alpha_{0123}t_6 \\ U_1(\mathbb{A}) =& \alpha_0 t_0 + \alpha_1 t_1 + \alpha_2 t_2 + \alpha_3 t_3 - \alpha_{012}t_3 \\ & - \alpha_{013}t_4 - \alpha_{023}t_5 - \alpha_{123}t_6, \end{aligned} \tag{9}$$

where α_i is a product of the corresponding elements in the micro-rotation direction vector, for instance, $\alpha_{012} = a_0a_1a_2$. Similarly, $\tau_{012} = t_0t_1t_2 = t_3$ due to the property that $t_it_j = t_{i+j}$.

B. Precalculation of the Rotation Vectors

The twiddle factors can be precomputed for all FFT applications. Likewise, the micro-rotation direction vectors in CORDIC-based FFT can also be precomputed in order to eliminate the need for recursions for calculating the micro-rotation matrices. Specifically, the rotation angles in FFT are not arbitrary but are incremented/decremented with a regular pattern, which also motivates the pre-calculation of the micro-rotation direction vectors. Eliminating recursions not only simplifies the circuit, but also reduces the power consumption of the circuit. Multiplication with negative one is the two's complement of that fixed point number. For example $a + (-1) * b$ can be done by passing the b operand inverted with carry input to the adder as one, thus eliminating the 2's complement operation needed for each micro-rotation stage. The ith micro-rotation stage engine shown in Fig. 2(a) does i left shifts of both the operands and performs an addition and subtraction. The complete unrolled non-recursive CORDIC is shown in Fig. 2(b).

C. Determining the Unrolling Factor

The unrolling factor needs to be determined depending upon the application and the angle quantization error that can be accommodated by the specific application. This error should not be mistaken as the computational noise introduced due to truncation and rounding in the fixed point computation blocks. This error is inherent to the CORDIC approximation of the FFT algorithm.

In order to evaluate the impact of the unrolling factor on the CORDIC-based FFT operation, the ideal DFT is used as a benchmark. A generic DFT operation is given in a matrix form by

$$\begin{bmatrix} X_{dft}(k) \\ Y_{dft}(k) \end{bmatrix} = \sum_{n=0}^{N-1} \begin{bmatrix} \cos(\frac{2\pi nk}{N}) & \sin(\frac{2\pi nk}{N}) \\ -\sin(\frac{2\pi nk}{N}) & \cos(\frac{2\pi nk}{N}) \end{bmatrix} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix}, \tag{10}$$

where N is the FFT size, $x(n)+jy(n)$ is the nth input complex sample, and $X_{dft}(k)+jY_{dft}(k)$ is the kth DFT output sample. On the other hand, the output of the proposed CORDIC-based

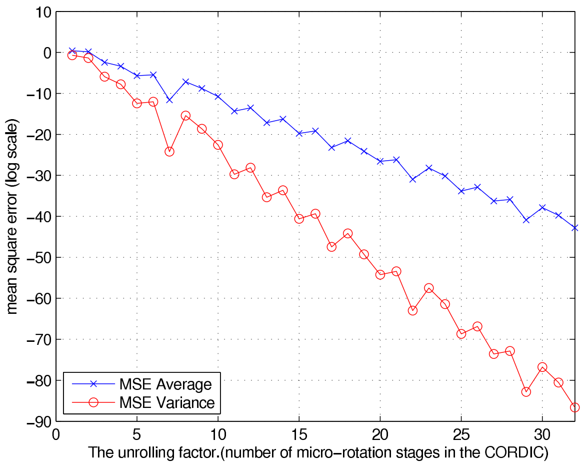


Fig. 3. Logarithm of mean square error ($\log_{10}(MSE)$) versus unrolling factor

FFT can be given by rewriting (6) as

$$\begin{bmatrix} X_p(k) \\ Y_p(k) \end{bmatrix} = \mathcal{C} \sum_{n=0}^{N-1} \begin{bmatrix} U_0(\mathbb{A}(n,k)) & U_1(\mathbb{A}(n,k)) \\ -U_1(\mathbb{A}(n,k)) & U_0(\mathbb{A}(n,k)) \end{bmatrix} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix}, \tag{11}$$

where $\mathcal{C} = \prod_{i=0}^{u} f_i \approx 0.60725$ and $\mathbb{A}(n,k)$ is the micro-rotation direction vector for the angle of $2\pi nk/N$. Notice that $\mathbb{A}(n,k)$ values can be pre-calculated and stored in a ROM table in order to ease the run-time processing.

Based on (10) and (11), the mean square error criterion is used with an error threshold to find the unrolling factor for the proposed CORDIC-based FFT as

$$\widehat{u} = \min_u \{\text{MSE}(u) \le \epsilon_{th}\}, \tag{12}$$

where ϵ_{th} is the mean square error threshold and

$$\text{MSE}(u) = \frac{1}{N} \sum_{k=0}^{N-1} [(X_{dft}(k) - X_p(k))^2 + (Y_{dft}(k) - Y_p(k))^2]. \tag{13}$$

In other words, the minimum unrolling factor is chosen among those providing a lower mean square error than the error threshold. Note that the threshold ϵ_{th} is determined based on the error margin that each application can tolerate. Fig. 3 shows the mean square error as a function of unrolling factor based on simulations.

IV. Complexity Comparison and FPGA Implementation

A. Computational Complexity for Radix-2, N-point FFT

The number of stages for a radix-2 FFT is $\log_2(N)$ where N is the size the radix-2 FFT. The number of stages that need true multiplication is $(\log_2(N) - 2)$. The number of adders per butterfly stage is 4 and the number of adders needed for a complex multiplication is 2. The total number of adders would be $(6\log_2(N) - 4)$, since the last two stages for a DIF

TABLE I

COMPARISON OF COMPUTATIONAL ELEMENTS.

	multipliers	adders
R2SD-FFT	$4(\log_2(N)-2)$	$6\log_2(N)-4$
R2SD-FFT (CORDIC)	None	$2(U+S)(\log_2(N)-2)+4\log_2(N)$

TABLE II

COMPUTATIONAL COMPLEXITY OF N-POINT RADIX-2 FFT ALGORITHM.

	multiplications	additions
R2SD-FFT	$2N(\log_2(N)-2)$	$3N\log_2(N)-4$
R2SD-FFT (CORDIC)	None	$N(U+S)(\log_2(N)-2)+2N\log_2(N)$

algorithm do not need multiplications and hence 4 adders can be removed from $6\log_2(N)$, while the total number of adders per stage is 6. The number of real multipliers per stage is 4.

Let S be the number of adders needed for the CORDIC scaling operation and U be the unrolling factor. There are 2 adders per micro-rotation. Then the adders needed for the unrolled CORDIC operation are $2(U+S)$. The total number of adders needed just for butterfly computations is $4\log_2(N)$. The total number of adders for the CORDIC FFT now becomes $(\log_2(N)-2)(2(U+S))$ + $4\log_2(N)$. The number of arithmetic blocks needed for computation is shown in Table I. The computational complexity can then be calculated by multiplying these values with N/r, where $r=2$ is the radix of the butterfly structure. Hence we have Table II for the computational complexity, where R2SD-FFT stands for radix-2 single delay feedback FFT. Notice that no multipliers are needed for the CORDIC based FFT. In order to make the comparison of the computational complexity of the CORDIC based FFT with the complex multiplier based FFT, we would have to equivalently represent the complex multiplier in terms of adder circuits. It can be easily seen that any B bit fixed point multiplication would require $B-1$ adders. Hence the total number of adders for the complex multiplier based FFT is $(6\log_2(N)-4)$ + $4B(\log_2(N)-2)$; notice that $N \geq 4$. Fig. 4 compares the complexity of the CORDIC FFT with the complex multiplier-based FFT for various FFT sizes.

B. Read Only Memory (ROM) Requirements

One of the key advantages of the CORDIC based FFT is the reduction of the twiddle factor memories. The twiddle factor memory for each stage depends upon three factors; arithmetic precision required for computation, number of stages in the DIT algorithm, and the number of points in the FFT. Assume B bits are used to represent each real and imaginary part of the twiddle factor, then we have $2B$ bits for a single twiddle factor. The 1st stage of the N-point FFT needs $N/2$ twiddle factors, the 2nd stage needs $N/4$ twiddle factors, etc. Usually data path implementation allows one bit of data growth every stage; hence to improve accuracy one bit of twiddle factor growth is also allowed. The twiddle factor memory size accounting for

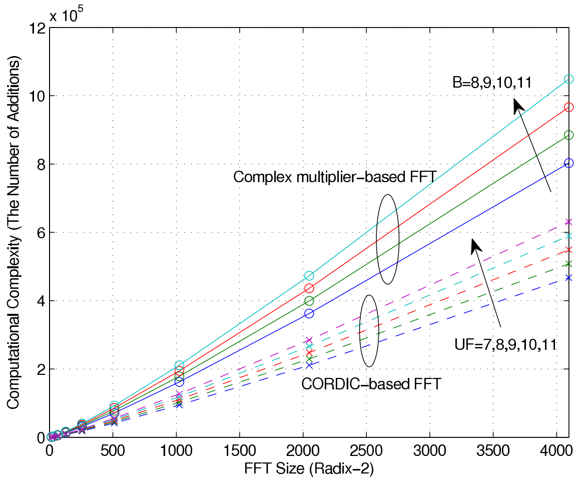


Fig. 4. Comparison of computational complexity

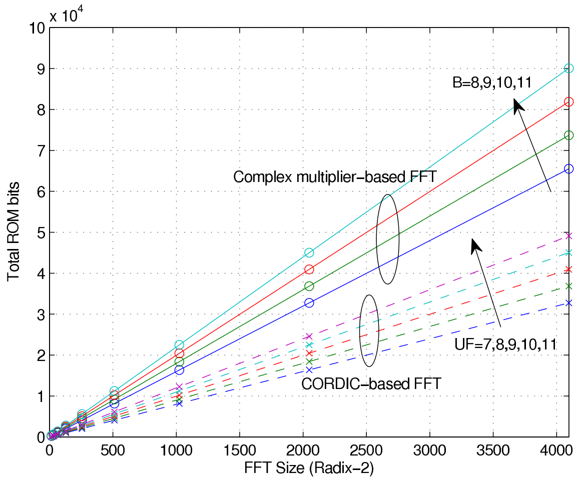


Fig. 5. Comparison of ROM table sizes

one bit growth is $2(B(N/2)+(B+1)(N/4)+(B+2)(N/8)+ ... +(B+log_2(N)-2))$, whereas twiddle factor memory size with no growth is $2B(N/2+N/4+N/8+...+2)$ = $2B(N-2)$ bits.

The rotation vector memory for a U unrolled CORDIC algorithm uses $(U+1)$ bits per rotation vector (one extra bit is need to indicate the quadrant information). Since the CORDIC can only compute rotations that lie in the range $-\pi/2$ to $\pi/2$, the total rotation vector memory requirement for the CORDIC based FFT would be $(U+1)(N-2)$. The general trend of the ROM requirements for R2SD-FFT and the R2SD-CORDIC-FFT is shown in Fig. 5.

C. On Accuracy of CORDIC based FFTs

Fig. 3 shows simulation results for 1024 point CORDIC based FFTs with various unrolling factors. As shown in Section III, the MSE provides vital information about the unrolling factor desired for CORDIC FFT implementation.

The simulation results point out that the error is reasonably low for an unrolling factor (UF) of seven. The MSE decreases with an increase in unrolling factor; however it is also observed that there is a local minimum at the unrolling factor of seven, with a slightly higher error for unrolling factors 8 and 9. However, this tapers down as seen in Fig. 3 for higher unrolling factors. In our case of $UF = 7$, the MSE is of the order of 10^{-11}. An interesting observation was found during the simulation; the unrolling factor is proportional to the logarithm of the MSE. Based on this observation, for any given applications a rigorous functional test algorithm can be derived to obtain the minimum unrolling factor that guarantees the MSE performance for the CORDIC based FFT.

D. Comparison of Area and Speed

We synthesized a complex multiplier based 1024 point FFT and a seven times unrolled CORDIC based 1024 point FFT, targeted to an FPGA board (Xilinx part number xc4vfx140ff1517-12). For fair comparisons, we prevented the use of the FPGA's internal DSP multipliers during the synthesis process. The area complexity inside the FPGA is based on the Xilinx look-up-table (LUT) count. Table IV shows the percentage savings on various FPGA resources.

TABLE III
FPGA TIMING REPORT.

	FFT1024_CORDIC	FFT1024
Max clock period	5.048ns(198.098 MHz)	19.707ns(50.74MHz)

TABLE IV
FPGA UTLILIZATION REPORT.

FPGA RESOURCE	FFT1024 _CORDIC	FFT1024 (complex multiplier)	% saving
BUF	3	3	0
FDC	374	354	-6
FDCE	26641	26640	0
GND	17	25	32
MULT_AND	582	2170	73
MUXCY_L	659	6420	90
MUXF5	2	21	90
VCC	2	10	80
XORCY	683	6212	89
MUXCY	0	149	100
LUT1	0	406	100
LUT2	26732	32617	18
LUT3	679	2907	77
LUT4	335	7141	95
I/O ports	69	69	0
I/O primitives	68	68	0
IBUF	25	25	0
OBUF	43	43	0
BUFGP	1	1	0
I/O Register bits	0	0	0
Register bits	27015	26994	0
Total LUTS	27746	44507	38

V. Conclusion and Future Work

A non-iterative CORDIC-based FFT is designed and evaluated in this paper. The unrolled CORDIC operations allows parallel processing as well as a pre-calculation for the micro-rotation matrix and the direction vector, which results in a high speed and high throughput FFT processor. The minimum unrolling factor was obtained from simulation. Since the micro-rotation stages use adder circuits which are not shared among the real and imaginary parts, it is further possible to optimize the design of micro-rotation stages at the logic level.

References

[1] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 5, pp. 911–919, May 2005.

[2] C.-Y. Yu, S.-G. Chen, and J.-C. Chih, "Efficient CORDIC designs for multi-mode OFDM FFT," in *Proc. IEEE Int. Conf. Acoustics, Speech and Sig. Processing*, vol. 3, May 2006, pp. 1036–1039.

[3] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-point Fourier Transform chip for high-speed Wireless LAN application using OFDM," *IEEE J. Solid-State Circuits*, vol. 39, no. 3, pp. 484–493, Mar. 2004.

[4] Z. Wang, M. Dong, and Y. Zhao, "Design and implementation of efficient FFT processor for multicarrier system," in *Proc. IEEE Canadian Cconference on Electrical And Computer Engineering*, May 2005, pp. 1384–1387.

[5] S. He and M. Torkelson, "A new approach to pipeline FFT processor architecture," in *Proc. IEEE Int. Parallel Processing Symp.*, Apr. 1996, pp. 766–770.

[6] H.-Y. Lee and I.-C. Park, "Balanced binary-tree decomposition for area-efficient pipelined fft processing," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 4, pp. 889–900, Apr. 2007.

[7] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar. 1999.

[8] J. H. Baek, B. S. Son, B. G. Jo, S. K. Oh, and M. H. Sunwoo, "A continuous flow mixed-radix FFT architecture with an in-place algorithm," in *Proc. IEEE Int. Symp. Circuits And Systems*, vol. 2, May 2003, pp. 133–136.

[9] L. Jia, Y. Gao, and H. Tenhunen, "Efficient VLSI implementaion of radix-8 FFT algorithm," in *Proc. IEEE Pacific Rim Conf. Commun. Computers and Sig. Processing*, Aug. 1999, pp. 468–471.

[10] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sept. 1959.

[11] A. M. Despain, "Fourier Transform computers using CORDIC iterations," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 993–1001, Oct. 1974.

[12] ——, "Very fast Fourier Transform algorithms hardware for implementation," *IEEE Trans. Comput.*, vol. C-28, no. 5, pp. 333–341, May 1979.

[13] I. Koren, *Computer Arithmetic Algorithms*. A. K. Peters, 2002.

[14] J. Duprat and J.-M. Muller, "The CORDIC algorithm : New results for fast VLSI implementation," *IEEE Trans. Comput.*, vol. C-42, no. 2, pp. 168–178, Feb. 1993.

[15] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Comput.*, vol. C-46, no. 8, pp. 855–870, Aug. 1997.

[16] T. B. Juang, "Area/delay efficient recoding methods for parallel CORDIC rotations," in *Proc. IEEE Int. Symp. Circuits And Systems*, June 2006, pp. 1539–1542.

[17] S. W. Mondwurf, "Versatile COFDM design based on the CORDIC-algorithm," *IEEE Trans. Consumer Electron.*, vol. 48, no. 3, pp. 718–723, Aug. 2002.

[18] B. S. Son, B. G. Jo, M. H. Sunwoo, and Y. S. Kim, "A high-speed FFT processor for OFDM systems," in *Proc. IEEE Int. Symp. Circuits And Systems*, vol. 3, May 2002, pp. 281–284.