# Enhanced Hardware Efficient FFT Processor based on Adaptive Recoding CORDIC

Jianfeng Zhang[1], Hengzhu Liu[1], Ting Chen[1], Dongpei Liu[1], Botao Zhang[1]
[1]*Computer School, National University of Defense Technology, Changsha, P.R. China*
*jianfengzhang@nudt.edu.cn*

***Abstract*—In this paper, we propose an enhanced hardware efficient CORDIC-based FFT processor. As the conventional CORDIC is restricted by the data precision and the times of iterations, Adaptive Recoding CORDIC (ARC) is adopted in our design, the precision of which is improved to 14th. Simultaneously, Conflict-free parallel memory access scheme and Rom-free twiddle factor generation scheme are both introduced to improve the performance and reduce the memories to store the twiddle factors. Compared with some latest published FFT processors, synthesized results show the proposed FFT processor reduce the hardware overhead while improving the Signal-to-Noise Ratio (SNR). When the operating frequency is 250MHz, the proposed FFT processor performs radix-4 1024-point FFT every 5.4 us.**

***Index Terms*—Fast Fourier transforms, CORDIC, bit error precision , signal to noise ratio.**

## I. INTRODUCTION

FFT processor, known as a specialized hardware, is indispensable for real-time signal processing and has been widely used in communication systems, such as WiMax [1] and 3GPP-LTE [2]. In this paper we focus on the efficient VLSI architecture with minimal hardware overhead and the high precision to compute FFT in real-time. Many people have researched on FFT processor design and implementation [3]–[6], which can be classified into two styles, pipelined and memory-based architectures. Due to speed acceleration, Pipelined architectures are widely used [3], [4]. Despite the attractiveness, Pipelined architectures have some drawbacks, such as excessive area and exhausted power consumption. Memory-based FFT processors are composed of a kernel processing unit and several memory blocks, the hardware requirement and power consumption of which are both lower than pipelined FFT processors [5], [6], and we adopt the memory-based FFT in our work.

The typical FFT processor is composed of butterfly units, address generator unit, control unit and memories. Butterfly units are composed of complex multipliers and adders. And one complex multiplier needs four real multipliers and two adders, thus the butterfly units are the speed bottleneck in FFT processor. An important statement for nowadays circuits design and implementation is hardware efficiency, and the coordinate rotation digital computer (CORDIC) algorithm perfectly fits this point. And in rotation mode, CORDIC algorithm can easily implement complex vector rotation, simply needs shift and addition operations, and it is feasible to be implemented by a pipelined method [7].

Besides, CORDIC-based FFT processors have been researched [5], [6], [8], and many of them adopt the conventional CORDIC, which heavily constraint the precision of complex multiplication. In this paper, we present an area-efficient pipeline-balancing CORDIC architecture, and in 16-bit computer, the Bit Error Position (BEP) of the proposed architecture has been improved with no performance penalty. Conflict-free parallel memory access scheme is adopted and ROM-free twiddle factor generator is proposed in our paper. The synthesized results state the proposed FFT processor requires less area and the Signal-to-Noise Ratio (SNR) is improved.

The following paper is organized as follows. Section II describes the top design of FFT processor. Adaptive recoding CORDIC-based butterfly is introduced in section III. In section IV, we present the simulation results and give some discussions. Summary of this work and conclusions are presented in section V.

## II. TOP DESIGN OF FFT PROCESSOR

FFT processor can use radix-2 algorithm, radix-4 algorithm, split-radix algorithm and so on. In this paper, we adopt the radix-4 algorithm. Fig. 1 shows the top 1024-point architecture of FFT processor, which is composed of control unit, address generate unit, twiddle factor angle generator, adaptive recoding CORDIC (ARC) based-butterfly unit, routing network, multiplexers (Mux) and memory banks. The control unit controls other FFT processor units. Through mux1, the input data are selected into mux2~5. Mux2~5 are used to select the signals passing to memory banks from the input data and the computed data after routing network. Address generate unit generates the write addresses and read addresses of memory banks. As the FFT processor is ROM-free twiddle factor, the twiddle factor angle generator controls generating the twiddle factors in real-time. Routing network is used to issue operands. The final results are the effective outputs of the mux6~9.

In the design of FFT processor, the conflict-free memory access is very important. Some scholars have proposed

Manuscript received March 28, 2012; accepted August 5, 2012.
This work was partly supported by National Natural Science Foundation of China (No.60970037) and Doctor Program Foundation of Education Ministry of China (No.20094307110009).

effective schemes. As [9] only needs bitwise XOR operations, we adopt the method and improve it in our radix-4 1024-point architecture. In our FFT processor, we only design one butterfly unit, thus four memory banks are required. Thus, the address of data includes two parts, two-bit bank address (Ba) and eight-bit depth address (Da). The signal Address-Num (AN) generated by control unit passes to the address generate unit. The bank address and depth address are defined as follows:

$$\begin{cases} Da = AN[9:2], \\ Ba[1] = AN[9]\text{^}AN[7]\text{^}AN[5]\text{^}AN[3]\text{^}AN[1], \\ Ba[0] = AN[8]\text{^}AN[6]\text{^}AN[4]\text{^}AN[2]\text{^}AN[0]. \end{cases} \quad (1)$$

Using the proposed conflict-free memory access scheme, the four operands of the butterfly unit would be distributed in different memory banks at the same time, the related distribution are shown in (2), which is a 4×4 matrix. The columns of the matrix stand for the related four situations, the rows are the corresponding number of the memory banks of the operands

$$Dis\ Matrix = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix}. \quad (2)$$

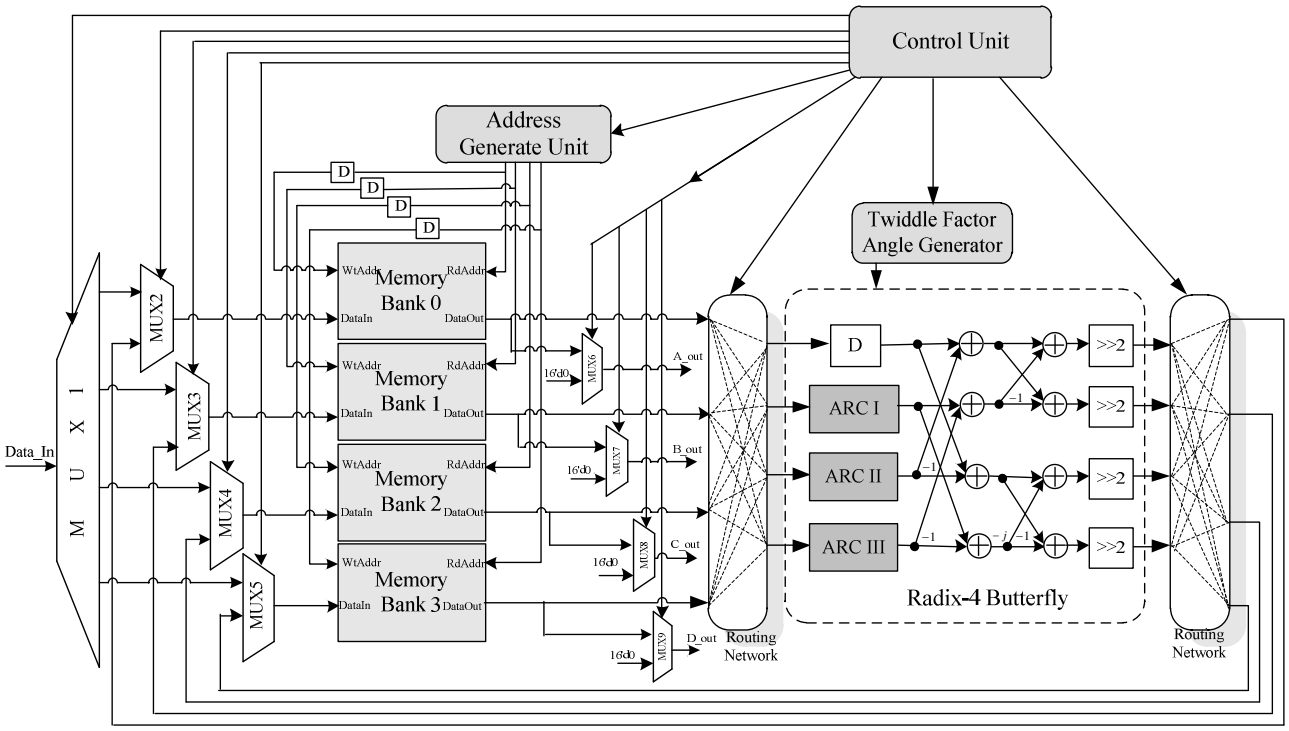



Fig. 1. Top level 1024-point architecture of FFT processor.

## III. ADAPTIVE RECODING CORDIC-BASED BUTTERFLY UNIT

The conventional Coordinate Rotation Digital Computer (CORDIC) algorithm was first brought forward to solve the trigonometric computing problem [7], simply needs shift and addition operations. However, it has some drawbacks, such as excessive number of iterations, the scaling factor and poor precision. Some works tried to enhance CORDIC [10], [11]. However, each of them either doesn't reduce the times of iterations or complicate the structure. In this paper, a novel architecture named adaptive recoding CORDIC (ARC) is adopted based on SF principle [10], which adopts adaptive recoding method to reduce the iterations, the complexity and improves the precision.

In FFT processor, the word length of the machine is configurable, for sake of clarity, the implementation of a 16-bit computer is described here as an example, and the complement of decimal 1 is represented as 16'b0100 0000 0000 0000, of which the most significant bit is the sign of data. When right shifting a data more than 14 bits, it will only leave the sign bits, which can be regarded as the machine zero.

As the convergence angle range of SF CORDIC is $\left[0, \pi/8\right]$, where $\pi/8 = 0.392699$ is represented as 16'b0001 1001 0010 0010, we only need deal with the 13 least significant bits of the angle. We divide the 13 bits into 2 groups, the first 3 bits belong to the first group, and the rest belong to the second group, as the lowest bit weight of the first group is $2^{-4}$, which is the upper limit of bit weight suiting for SF CORDIC. The second group is divided into 5 sub-groups, and each sub-group contains two neighbouring bits. Our aim is to make each sub-group only consists one effective bit, which means the iterations can be reduced from 2 to 1.

According to the representation of complement, the bit weights of two neighbouring bits satisfy:

$$\begin{cases} 2^{-i}+2^{-i-1}=2^{-i+1}+0\times 2^{-i}+(-1)\times 2^{-i-1} \\ 2^{-i}+0\times 2^{-i-1}=2^{-i+1}+(-1)\times 2^{-i}+0\times 2^{-i-1} \end{cases} \quad (3)$$

This means if the neighbouring bits are 11 or 10, they could be recoded to 01 or 10, the symbols are changed into 01 or 10, while generating a carry bit. Different from SF CORDIC, which only rotates in anti-clockwise direction, our scheme allows rotating in clockwise direction, which is determined by the symbols of the recoded bits. If the symbol of the bit is 1, it rotates in clockwise direction; otherwise it would rotate in anti-clockwise direction. Fig. 2(a) shows the rules of this recoding. The circuit of fundamental recoding unit is shown in Fig. 2(b). The main components of fundamental recoding unit are two AND gates and one inverter. The higher input bit is executed by the operation AND with the lower bit and its inverse signal to get the recoded bits, symbol bits and carry bit. Fig. 2(c) illustrates the whole architecture of adaptive recoding, which consists of five fundamental recoding units, five adders and five OR gates. In Fig. 2(b), "Module #i" has 5 bits outputs, while "Module #2", "Module #3", "Module #4" and "Module #5" seem to have 3 output signals, because we synthesize the two bits of recoded data "R_M$_{\#i}$[1:0]" and the two bits of the symbol of the recoded data "S_M$_{\#i}$[1:0]" in Fig. 2(c), as the carry bit would pass to the higher level module, but the recoded data, the symbol and the carry bit of lower level module need to be operated by the adder to get the final data out and the final symbol of the data out. And in Fig. 2(c), as the "Module #1" is the lowest level of module, there is no carry bit pass to it, but the nearby two bits of recoded data need to be operated to get the enable signal. Thus, we just synthesize the symbol of the recoded data. In order to better realize this and not generate various understandings, we have marked the bits of the outputs in Fig. 2(c). Firstly, the five sub-groups pass through the five fundamental recoding units separately, and then the carry bits are sent to the higher adders and the last carry bit of the fifth recoding unit passes to the first group. Finally, the operated signals pass through OR gates, then we would get the recoded data, symbol bits and control signals.

Theorem: If each sub-group is recoded according to the principle of adaptive recoding, that its output only contains one effective value regardless of the number of sub-groups.

Proof: Set:

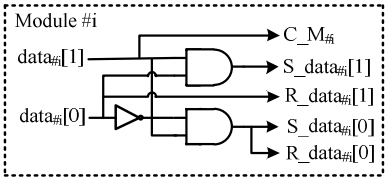
$$\begin{cases} \phi=\{First\_group, Second\_group\}, \\ First\_group=\{F\_data[2:0]\}, \\ Second\_group=\{M_{\#n},\cdots,M_{\#i},\cdots,M_{\#2},M_{\#1}\}. \end{cases} \quad (4)$$

In which $M_{\#i}=\{data_{\#i}[1:0]\}$ indicates the i$^{th}$ sub-group in the second group, and the maximum value of first group is 110, which is restricted by the convergence range of SF theory. According to the circuit shown in Fig. 2.(b), the detailed expressions can be defined as follows:

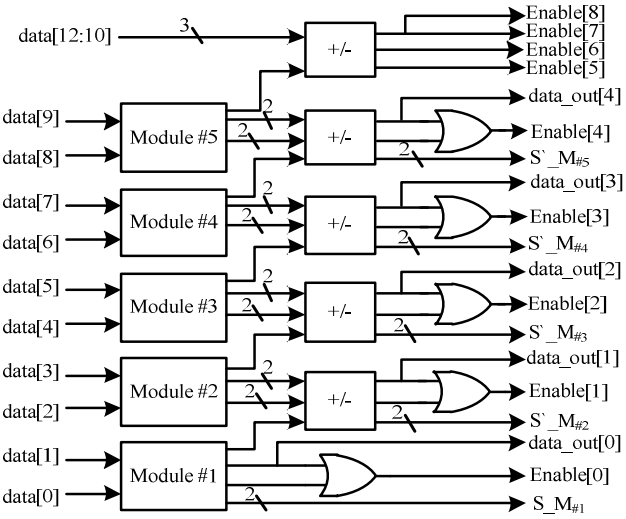
$$\begin{cases} R\_M_{\#i}[1]=data_{\#i}[1]\,\&\,\overline{data_{\#i}[0]}, \\ R\_M_{\#i}[0]=data_{\#i}[0], \\ R\_M_{\#i}=\{R\_M_{\#i}[1],R\_M_{\#i}[0]\}, \\ S\_M_{\#i}[1]=data_{\#i}[1], \\ S\_M_{\#i}[0]=data_{\#i}[1]\,\&\,data_{\#i}[0], \\ S\_M_{\#i}=\{S\_M_{\#i}[1],S\_M_{\#i}[0]\}, \\ C\_M_{\#i}=data_{\#i}[1]. \end{cases} \quad (5)$$

| Former Code | Recode Code | Recode Symbol | Recode Carry bit |
|---|---|---|---|
| 00 | 00 | 00 | 0 |
| 01 | 01 | 00 | 0 |
| 10 | 10 | 10 | 1 |
| 11 | 01 | 01 | 1 |

(a)




(b)




(c)

Fig. 2. Principle and architecture of adaptive recoding: (a) – principle of adaptive recoding; (b) – circuit of the fundamental recoding unit; (c) – architecture of adaptive recoding.

Thus, adopting the complete induction to prove the theorem as follows:

a) Assuming that $Second\_group=\{M_{\#1}\}$, and $M_1=\{data_{\#i}[1],data_{\#i}[0]\}$, according to (5):

$$\begin{cases} M_{\#1}=00\Rightarrow R\_M_{\#1}=00, S\_M_{\#1}=00, C\_M_{\#1}=0, \\ M_{\#1}=01\Rightarrow R\_M_{\#1}=01, S\_M_{\#1}=00, C\_M_{\#1}=0, \\ M_{\#1}=10\Rightarrow R\_M_{\#1}=10, S\_M_{\#1}=10, C\_M_{\#1}=1, \\ M_{\#1}=11\Rightarrow R\_M_{\#1}=01, S\_M_{\#1}=01, C\_M_{\#1}=1. \end{cases} \quad (6)$$

Thus $R\_First\_group=\{F\_data[2:0]+C\_M_{\#1}\}$, and $R\_Second\_group=\{R\_M_{\#1}\}$, so the lemma is right.

b) Assuming that $Second\_group=\{M_{\#(k-1)},\cdots,M_{\#2},M_{\#1}\}$, then $R\_Second\_group=\{R'\_M_{\#(k-1)},\cdots,R'\_M_{\#2},R\_M_{\#1}\}$, $R\_First\_group=\{F\_data[2:0]+C\_M_{\#(k-1)}\}$,and $R'\_M_{\#i}$ means the final recoded data accepting the lower level carry bit, and the lemma is right. Then when $Second\_group'=\{M_{\#k},Second\_group\}$. Firstly, we recode the added module $M_{\#k}$ according to (5):

$$\begin{cases} M_{\#k}=00\Rightarrow R\_M_{\#k}=00,S\_M_{\#k}=00,C\_M_{\#k}=0,\\ M_{\#k}=01\Rightarrow R\_M_{\#k}=01,S\_M_{\#k}=00,C\_M_{\#k}=0,\\ M_{\#k}=10\Rightarrow R\_M_{\#k}=10,S\_M_{\#k}=10,C\_M_{\#k}=1,\\ M_{\#k}=11\Rightarrow R\_M_{\#k}=01,S\_M_{\#k}=01,C\_M_{\#k}=1. \end{cases} \quad (7)$$

Then $R\_First\_group'=\{F\_data[2:0]+C\_M_{\#k}\}$, considering the carry bit $C\_M_{\#(k-1)}$ generated by $M_{\#(k-1)}$, if $C\_M_{\#(k-1)}\neq 0$, then it needs pass to $M_{\#k}$. And if $S\_M_{\#k}=00$, $M_{\#k}$ will add the carry bit, whereas $M_{\#k}$ will subtract it, since the symbol $S\_M_{\#k}$ is negative, which indicates that the conventional addition will change into subtraction. Hence:

$$\begin{cases} R\_M_{\#k}=00,S\_M_{\#k}=00\Rightarrow R'\_M_{\#k}=01,S'\_M_{\#k}=00,\\ R\_M_{\#k}=01,S\_M_{\#k}=00\Rightarrow R'\_M_{\#k}=10,S'\_M_{\#k}=00,\\ R\_M_{\#k}=10,S\_M_{\#k}=10\Rightarrow R'\_M_{\#k}=01,S'\_M_{\#k}=01,\\ R\_M_{\#k}=01,S\_M_{\#k}=01\Rightarrow R'\_M_{\#k}=00,S'\_M_{\#k}=00, \end{cases} \quad (8)$$

and $R\_Second\_group'=\{R'\_M_{\#k},R\_Second\_group\}$, as previous description, each sub-group in the second group only contains one effective value. Thus, the lemma is right.

c) According to the step a) and b), we can conclude that no matter how many sub-groups does the second group have, the lemma works right.

As the bit weight of the first bit in first group is $2^{-2}$, which needs repeat the conventional i=4 SF unit four times, and the second bit needs repeat twice. Eq. (7) gives the matrix of conventional i=4 SF unit:

$$\begin{bmatrix} x'\\ y' \end{bmatrix}=\begin{bmatrix} 1-2^{-9} & -2^{-4}\\ 2^{-4} & 1-2^{-9} \end{bmatrix}\begin{bmatrix} x\\ y \end{bmatrix}. \quad (9)$$

Then the execution time of i=4 SF unit lies on one shift and two additions operations. Whereas, if multiply the matrix of i=4 twice, it can be easily verified that the enhanced matrix is:

$$\begin{bmatrix} x'\\ y' \end{bmatrix}=\begin{bmatrix} 1-2^{-7} & -(2^{-3}-2^{-12})\\ 2^{-3}-2^{-12} & 1-2^{-7} \end{bmatrix}\begin{bmatrix} x\\ y \end{bmatrix}. \quad (10)$$

Hence, the execution time of (10) is the same as (9) except for required more shifters and adders. By the enhanced matrix (10), the first bit in first group only needs repeat the matrix process twice, and the second bit needs once, while the last bit still executes the process (9).

In the proposed scheme, we do some minor modification based on SF architecture. Since the theory of SF magnifies the trigonometric value, we scale down the cosine value of the first enhanced matrix into ($2^{-3}$-$2^{-11}$) as compensation after self-contained test to improve the precision.

Fig. 3 shows the whole pipeline of adaptive recoding CORDIC, the hardware block named "Con. i. Unit" indicates the conventional SF module, and i means the bit weight of the module is $2^{-i}$. As above discussion, the bits whose bit weight less than $2^{-4}$ are grouped into the second group, and each sub-group in which would only exist one effective value after adaptive recoding, while the execution time of the conventional i=5/6 SF unit still lies on one shift and two additions. But (i=7/8, i=9/10) and (i=11/12, i=13/14) modules are integrated into a single stage to balance the latency of every stage, because the execution time of the mentioned modules lie on one shift and one addition operations. In Fig. 3, the pipeline stage registers are given with different colors. Firstly, the original signals are sent to the pre-processing and recoding module, the control signals, recoded angle and other signals will be achieved after recoding. Secondly, the signals will be handled by the following modules or bypass according to the enable signals. At last, the post-processing module completes the conversion of trigonometric function (from $\left[0,\pi/8\right]$ to $\left[0,2\pi\right]$).

The expression of complex multiplication can be defined as follows:

$$\begin{aligned} z(k)\cdot W_N^{nk} &= [real(z)+j\cdot imag(z)]\cdot[\cos(-\frac{nk}{N}2\pi)+j\cdot\sin(-\frac{nk}{N}2\pi)]=\\ &= real(z)\cdot\cos(-\frac{nk}{N}2\pi)-imag(z)\cdot\sin(-\frac{nk}{N}2\pi)+\\ &\quad + j\cdot[real(z)\cdot\sin(-\frac{nk}{N}2\pi)+imag(z)\cos(-\frac{nk}{N}2\pi)]. \end{aligned} \quad (11)$$

The primary rotation matrix of the CORDIC can be described as (considering anti-clockwise rotation):

$$\begin{bmatrix} x'\\ y' \end{bmatrix}=\begin{pmatrix} \cos\theta & -\sin\theta\\ \sin\theta & \cos\theta \end{pmatrix}\cdot\begin{bmatrix} x\\ y \end{bmatrix}, \quad (12)$$

where $\theta$ is the target angle of the rotation. Assuming that $\theta=-\frac{nk}{N}\cdot 2\pi$, $x=imag(z)$, $y=real(z)$, then $x'$ is the imag result of complex multiplication, and $y'$ is the real result of complex multiplication.

Conventional FFT processors need a large ROM to store the twiddle factors. During the process of butterfly operations, we need issue more load instructions to fetch the twiddle factors, which are slow speed, large area and high power consumption. In our paper, we propose a novel Rom-free twiddle factor generation scheme, then the proposed processor needs not to store twiddle factors anymore, which

would be generated in real-time. During one radix-4 butterfly operation, it needs three twiddle factors, which are $W_{1024}^{k}$, $W_{1024}^{2k}$, $W_{1024}^{3k}$. If we get know the twiddle factor $W_{1024}^{k}$, the others are all known. $W_{1024}^{k} = e^{-j\frac{k}{1024}\cdot 2\pi}$, the angle of the twiddle factor is $\theta = 2k\pi/1024$. If k is got, it just needs to be scaled by $2\pi/1024$ to get the angle. The complement of $2\pi/1024$ is 16'b0000 0000 0110 0100, thus $\theta = k\cdot(2^2+2^5+2^6)$. As the radix-4 1024-point FFT processor needs 5 butterfly pipeline levels, the value of k changes with levels, which are shown in Table1. And the 8-bit signal cycle num (CN), generated by the control unit.

TABLE I. BASIC INDEX OF THE ANGLE OF THE TWIDDLE FACTOR.

| level | BasicIndex($\theta$) |
|---|---|
| 0 | 16'd0 |
| 1 | {CN[1:0], 2'd0}+{CN[1:0], 5'd0}+{CN[1:0], 6'd0} |
| 2 | {CN[3:0], 2'd0}+{CN[3:0], 5'd0}+{CN[3:0], 6'd0} |
| 3 | {CN[5:0], 2'd0}+{CN[5:0], 5'd0}+{CN[5:0], 6'd0} |
| 4 | {CN, 2'd0}+{CN, 5'd0}+{CN, 6'd0} |

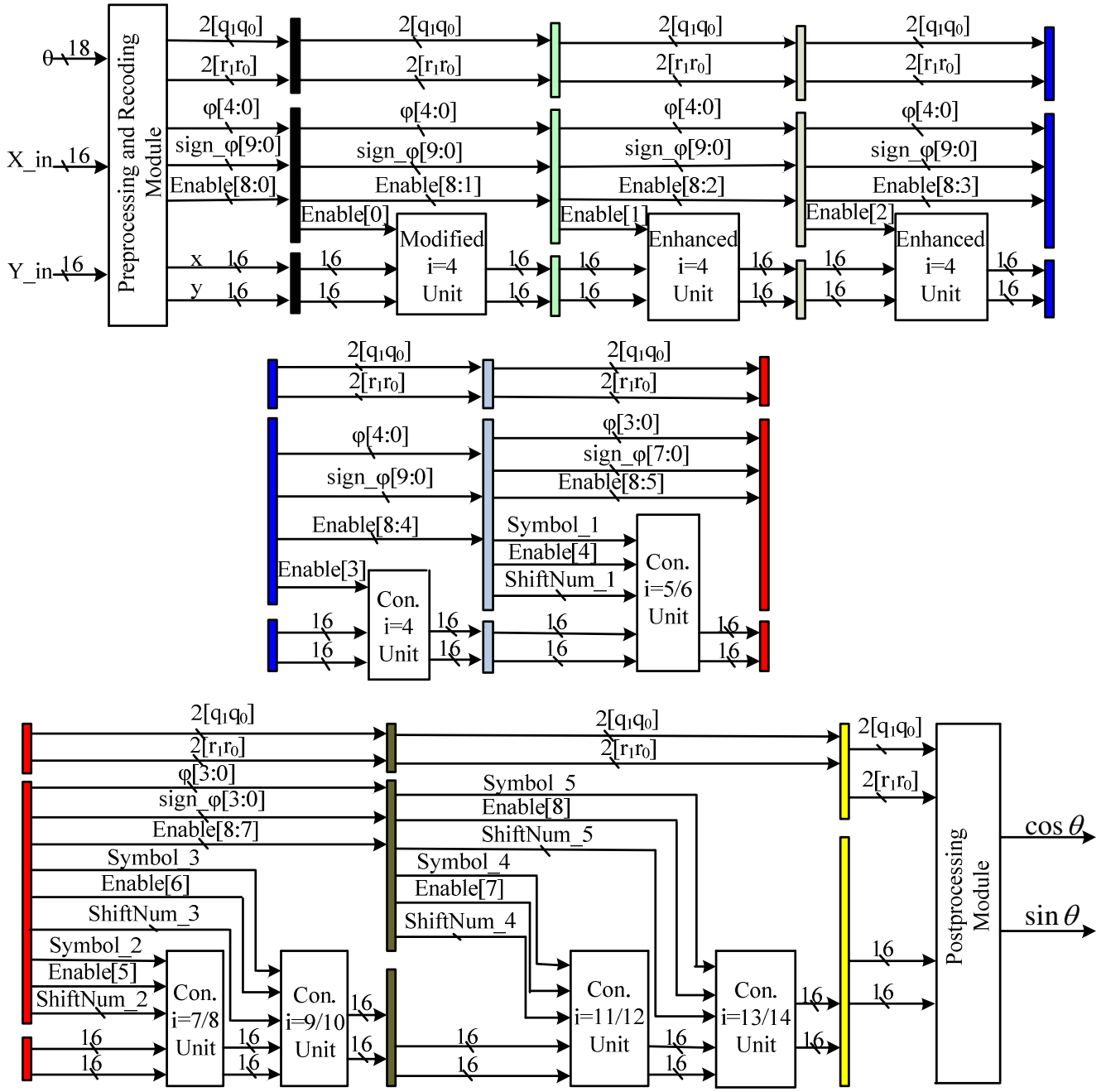



Fig. 3. The whole pipeline of Adaptive Recoding CORDIC.

Then the proposed Rom-free twiddle angle generation scheme only needs adders and shifters. Eq. (13) gives the input angles of the three ARC units:

$$\begin{cases} ARCI\_\theta = \theta, \\ ARCII\_\theta = \{\theta, 1'b0\}, \\ ARCIII\_\theta = ARCI\_\theta + ARCII\_\theta. \end{cases} \quad (13)$$

## IV. PERFORMANCE EVALUATION AND COMPARISON

The errors in CORDIC algorithm mainly come from quantization process and truncation process [12]. We analyze the error of the proposed ARC algorithm by comparing Bit Error Position (BEP). The expression is described as

$$BEP = -\log_2 \left| x_c - x_r \right|, \tag{14}$$

where $x_c$ is the value computed by the proposed algorithm, and $x_r$ is the value by looking up table which stores the real value of the trigonometric function. We generate a pseudorandom sequence of angles lying within the convergence range $[0, \pi/8]$. Using these angles as the inputs, the corresponding errors in terms of the decimal bit position are shown in Fig. 4.

The error of adaptive recoding CORDIC locates above the 14th decimal bit position for all cases, which is the same as SFB4C [11], and exceeds conventional SF CORDIC two bits [10].

The proposed ARC-based FFT is modeled in Verilog HDL and synthesized with the Synopsis Design Complier based on Chartered 90nm CMOS technology standard cell library. Table2 shows the area and speed comparison of the proposed FFT processor with some latest published ones. The execution time of FFT is defined as

$$T_{exec} = \frac{1}{f_w}\left[ \left(\log_4 N\right) \cdot \left( \frac{N}{4} + D_L \right) \right] \tag{15}$$

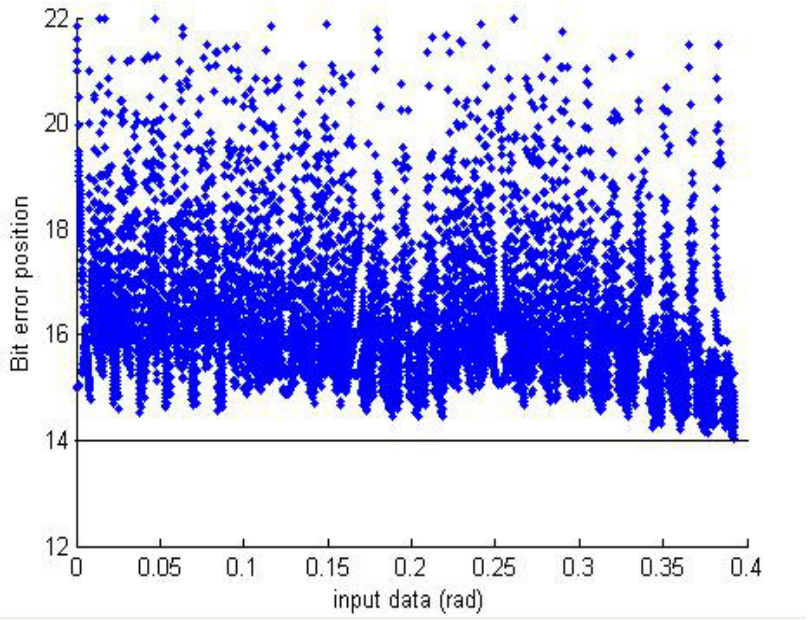



Fig. 4. Accuracy Cosine errors for the proposed ARC.

TABLE II. COMPARISON OF THE AREA AND SPEED.

| | [13] | [14] | Proposed |
|---|---|---|---|
| **Tech. (nm)** | 130 | 130 | 90 |
| **Area ($mm^2$)** | 2.96 | 3.75 | 1.09 |
| **1024-pt Exec.Time** | 51.6μs (250MHz) | 21.0μs (250MHz) | 5.4μs (250MHz) |

In which $f_w$ is the operation frequency of the FFT processor, N is the FFT size, and $D_L$ is the set up stages of the whole pipeline. The execution time of the proposed FFT processor performs 1024-point FFT every 5.4us under 250 MHz frequency. In Table2, it can be seen the proposed FFT processor reduces hardware overhead and the execution process is accelerated, owing to the pipelined adaptive recoding CORDIC based radix-4 FFT structure without multipliers and Rom-free twiddle generator needless Rom to store twiddles.

Owing to the finite word-length implementation, the outputs of FFT processor exist truncation error and quantization error. In order to evaluate the Signal-to-Noise Ratio (SNR), we generate a random noise signal as the inputs of FFT processor by Matlab and get the results $D_{Mod}$ based on Modelsim simulation. Simultaneously, the random noise signal is utilized by a Matlab script to get the double FFT results $D_{Mat}$.

And we use the (16) to evaluate the SNR of the proposed FFT processor. The results show the SNR of the proposed FFT processor is 42dB, which almost exceeds conventional CORDIC-based FFT 7dB and the FFT processor proposed in [15] 5dB with 14 effective bits to avoid the overflow of radix-4 butterfly

$$SNR(dB) = 10 \cdot \log_{10}\left[ \frac{\sum_{i=0}^{N-1} \left| D_{Mat} \right|^2}{\sum_{i=0}^{N-1} \left| D_{Mat} - D_{Mod} \right|^2} \right]. \tag{16}$$

## V. CONCLUSIONS

This paper proposed an enhanced hardware efficient FFT processor based on CORDIC. As restricted by the lower precision of conventional CORDIC, we adopted adaptive recoding CORDIC, the BEP of which was improved to 14th. In the proposed FFT processor, conflict-free parallel

memory access scheme is adopted and Rom-free twiddle factor generation principle is introduced. The proposed FFT processor had been synthesized using 90nm CMOS technology with standard cell library, and the results show it is area efficiency, high SNR and less execution time.

## REFERENCES


[1] J. G. Andrews, A. Ghosh, R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Prentice Hall, 2007, p. 16.

[2] E. Dahlman, *3G Evolution: HSPA and LTE for Mobile Broadband.* Academic press, 2008, p. 18.

[3] M. Garrido, K. K. Parhi, J. Grajal, "A Pipelined FFT Architecture for Real-Valued Signals", *IEEE Transactions on Circuits and Systems–I: Regular Papers,* IEEE, vol. 12, no. 56, pp. 2634–2643, Dec. 2009.

[4] B. Zhou, Y. Peng, D. Hwang, "Pipeline FFT Architectures Optimized for FPGAs", *International Journal of Reconfigurable Computing,* Hindawi, pp. 1–9, 2009.

[5] X. Xiao, E. Oruklu, J. Saniie, "Reduced Memory Architecture for CORDIC–based FFT", in *Proc. of 2010 IEEE International Symposium on Circuits and Systems,* IEEE, 2010, pp. 2690–2693.

[6] E. Oruklu, X. Xiao, J. Saniie, "Reduced Memory and Low Power Architecture for CORDIC–based FFT Processors", *Journal of Signal Processing Systems,* Springer, vol. 2, no. 66, pp. 129–134, 2011.

[7] J. E. Volder, "The CORDIC trigonometric computing technique", *IEEE Transactions on Electron. Comput.*, IEEE, vol. 3, no. 8, pp. 330–334, Sept. 1959.

[8] T. Y. Sung, H. C. Hsin, Y. P. Cheng, "Low–power and High–speed CORDIC–based Split-radix FFT Processor for OFDM Systems", *Digital Signal Processing,* Elsevier, vol. 2, no. 20, pp. 511–527, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.dsp.2009.08.008

[9] J. H. Takalala, T. S. Jarvinen, H. T. Sorokin, "Conflict–free Parallel Memory Access Scheme for FFT Processors", in *Proc. of IEEE International Symposium on Circuits and Systems,* IEEE, 2003, pp. 524–527.

[10] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, "Modified virtually scaling–free adaptive CORDIC rotator algorithm and architecture", *IEEE Transactions on Circuits and Systems*, IEEE, vol. 11, no. 15, pp. 1463–1474, 2005.

[11] J. F. Jaime, A. M. Sanchez, "Enhanced Scaling–Free CORDIC", *IEEE Transactions on Circuits and Systems*, IEEE, vol. 7, no. 57, pp. 1654–1662, 2010. [Online]. Available: http://dx.doi.org/10.1109/TCSI.2009.2037391

[12] Y. H. Hu, "The quantization effects of the CORDIC algorithm", *IEEE Transactions on Signal Process,* IEEE, vol. 4, no. 40, pp. 834–844, 1992. [Online]. Available: http://dx.doi.org/10.1109/78.127956

[13] Q. Zhang, N. Meng, "A Low Area Pipelined FFT Processor for OFDM–Based Systems", in *Proc. of International Conference on Wireless Communications, Networking and Mobile Computing,* IEEE, pp. 1–4, 2009.

[14] T. Pitkanen, J. Takala, "Low–Power Application–Specific Processor for FFT Computations", *Journal of Signal Processing Systems,* Springer, pp. 165–176, 2011.

[15] A. Cortés, I. Vélez, A. Irizar, "An Approach to Simplify the Design of IFFT/FFT Cores for OFDM Systems", *IEEE Transactions on Consumer Electronics,* IEEE, vol. 1, no. 52, pp. 26–32, 2006. [Online]. Available: http://dx.doi.org/10.1109/TCE.2006.1605021