Vietnam Journal of Science and Technology ***56*** *(6) (2018) 751-764*

DOI: 10.15625/2525-2518/56/6/12269

DESIGN OF A HIGH-SPEED HIGH-ACCURACY 2048-POINT FFT USING SINGLE-PRECISION FLOATING-POINT ADAPTIVE CORDIC ON FPGA

Nhu-Quynh TRUONG[1], Trong-Thuc HOANG[2], Cong-Kha PHAM[2], Duc-Hung LE[1, *]

[1]*The University of Science, Vietnam National University Ho Chi Minh City,227 Nguyen Van Cu, District 5, Ho Chi Minh City, Viet Nam*

[2]*The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, 182-8585, Tokyo, Japan*

[*]Email: *ldhung@hcmus.edu.vn*

Received: 10 April 2018; Accepted for publication: 17 June 2018

ABSTRACT

In this paper, hardware design of a Fast Fourier Transform (FFT) core using Single-precision Floating-point Adaptive CORDIC is implemented on Altera Stratix IV FPGA. With FFT implementation, CORDIC is utilized for reducing the speed drawback of complex multiplication and the adaptive algorithm is proposed to decrease the iterations of conventional CORDIC. The experimental results of Adaptive CORDIC and 2048-point Radix-2 Multi-path Delay Commutator FFT designs are built and verified based on three kinds of Look-up Table that cost 16, 8 and 4 constant angles. As experimental results, there is a resource equivalence while it has a trade-off between speed performance and accuracy. In comparison, an adaptive CORDIC core based on Look-up Table of 16 constant angles, and 2048-point Radix-2 Multi-path Delay Commutator Fast Fourier Transform based on Adaptive CORDIC using Look-up Table of 16 constant angles are well responding to resource optimization, high-speed performance and high-accuracy of computations.

Keywords: adaptive CORDIC, FFT, floating-point, single-precision, high-accuracy.

Classification numbers: 4.1.1, 4.2.3, 4.9.3

1. INTRODUCTION

Fast Fourier Transform (FFT) is a fast computation method of Discrete Fourier Transform algorithm, firstly introduced by Cooley and Tukey in 1965 [1]. Until now, FFT and inverse FFT has been widely used in various applications of signal processing and communication systems. These applications include wireless and multimedia as MIMO [2], CDMA [3], WiMAX [4], 3GGP-LTE [5] or in the Orthogonal Frequency Division Multiplexing (OFDM) [6]. FFT

algorithm can be divided into two types such as fixed-point FFT and floating-point FFT. Fixed point FFT has the advantages of high speed and resource optimization. However, it has a high error-ratio by cutting out Least Significant Bits (LSBs) in computation. While nowadays, more and more advanced applications such as Synthetic Aperture Radar (SAR) [7] or medical image processing like Fourier-Domain Optical Coherence Tomography (FD-OCT) [8], require not only a high precision but also a wide dynamic range of data. As a result, the floating point becomes an essential requirement for computing large points of FFT. In terms of Floating point FFT, it can be classified into two primary architectures including Memory-based FFT [9] and Continuous Flow FFT [10]. The logic resource of Mem-based FFT is small, but slow processing speed due to high-latency performance. While Continuous Flow FFT becomes more common than Memory-based FFT because of high-throughput performances, its drawback is the high resource cost. Even any architectures, the speed bottleneck is Butterfly Unit or Complex Multiplication.

In this research, COordinate Rotation DIgital Computer (CORDIC) is proposed as a complex multiplication. Since being proposed more than 50 years, CORDIC is still one of the most effective algorithms for elementary function calculations. The conventional CORDIC has a great performance in fixed-point FFT because it reduces the complex multiplication into several simple shifts and additions. However, it is not suited for floating-point FFT because a floating-point addition is not easy to be implemented and the constant number of iterations. That will cause a high-cost and low throughput. Therefore, Adaptive CORDIC is proposed in this paper. The idea of Adaptive CORDIC is to select only a few angles to be rotated instead of all angles. Therefore, the system will be faster with an approximation accuracy result. Based on the previous works such as [11, 12], in this study, Adaptive CORDIC has been optimized and implemented for a Radix-2 Multi-path Delay Commutator FFT system in hardware as an application. In this research, the experimental results of Adaptive CORDIC and 2048-point Radix-2 Multi-path Delay Commutator FFT designs is built and verified based on three kinds of Look-up Table that cost 16, 8 and 4 constant angles. In comparison with these designs, this paper draws which is the most relevant number of constant angles to achieve the high accuracy but guarantee the optimizing speed.

The content of this paper is constructed as follows. Section 2 gives a brief review of Fast Fourier Transform and CORDIC methods. Section 3 presents the proposed design implementation. Section 4 provides the experimental results. And some discussions are given in Section 5 before concluding in Section 6.

2. RELATED WORKS

2.1. Fast Fourier Transform

Fast Fourier Transform (FFT), which is a fast computation method of Discrete Fourier Transform (DFT), has the similar method and result. However, their difference is the complexity of computing FFT as $Nlog_2(N)$ while DFT requires the complexity of N^2. The equation of FFT is shown in Eq. (1).

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

where $\boldsymbol{x(n)}$ is the input value in time domain, $\boldsymbol{X(k)}$ is the output value in frequency domain, and $\boldsymbol{W_N^{kn}}$ stands for Twiddle factor as complex multiplication in FFT's Butterfly unit, shown in Eq. (2).

$$W_N^{kn} = \exp\left(-\frac{j2\pi nk}{N}\right) = \cos\left(\frac{2\pi nk}{N}\right) - j\sin\left(\frac{2\pi nk}{N}\right) \quad (2)$$

2.2. Radix-2 Fast Fourier Transform

Radix-2 is a conventional Fast Fourier Transform algorithm. Due to its simplicity, Radix-2 has a low-cost resource. The idea of Radix-2 is that divides the set input data of N points into independently calculated $N/2$ even points $X(2r)$ and $N/2$ odd points $X(2r+1)$. With Radix-2 algorithm, the FFT formula which is described by Eq. (1), becomes the equivalent one as Eq. (3). And an example of 8-point FFT Radix-2 is shown in Fig. 1.

$$\begin{aligned} X(2r) &= \sum_{n=0}^{\frac{N}{2}-1} x(n) + x\left(n+\frac{N}{2}\right) W_{N/2}^{rn} \\ X(2r+1) &= \sum_{n=0}^{\frac{N}{2}-1} x(n) - x\left(n+\frac{N}{2}\right) W_N^n W_{N/2}^{rn} \end{aligned} \quad (3)$$

$$r = 0, 1, \ldots, \frac{N}{2} - 1$$

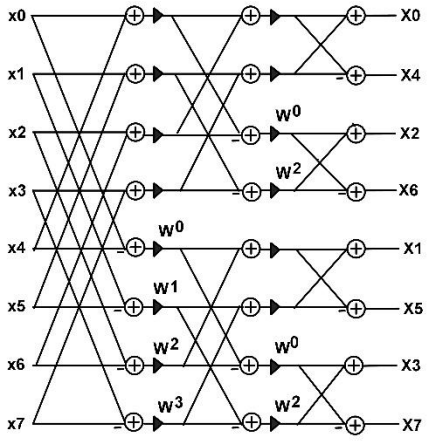


Figure 1. An example of 8-point Radix-2 FFT.

2.3. Multi-path Delay Commutator Fast Fourier Transform

Figure 2 presents the concept of Multi-path Delay Commutator (MDC). With this architecture, each stage includes two data buffers which the second buffer (Shift_regs2) has its capacity as half as the first ones (Shift_regs1). In the Radix-2 MDC design, after the input data is stored Shift_regs1, Butterfly computes this data and releases two data paths. One data path will directly obtain the final output, while another implements the complex multiplication and stores in Shift_regs2 before the final output.

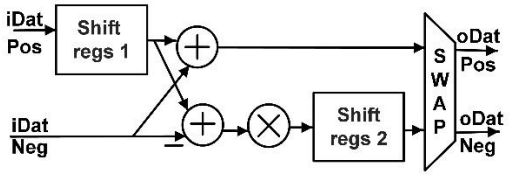


Figure 2. Multi-path Delay Commutator architecture.

2.4. Adaptive CORDIC

CORDIC, COordinate Rotation DIgital Computation, an efficient iterative algorithm to compute trigonometric and hyperbolic functions, includes two modes as rotation and vectoring mode. CORDIC algorithm needs several addition and shift operations described by the Eq. (4).

$$\begin{aligned} x_{i+1} &= x_i - s_i y_i 2^{-i} \\ y_{i+1} &= y_i + s_i x_i 2^{-i} \\ z_{i+1} &= z_i - s_i \alpha_i \\ s_i &= sign(z_i) \end{aligned} \tag{4}$$

where, x_{i+1} and y_{i+1} are the new coordinate values of the vector after rotating a new angle z_{i+1}, and α_i is called residual angle and computed by the equation Eq. (5).

$$\alpha_i = \tan^{-1} 2^{-i} \tag{5}$$

in each iteration, the axis values are increased by a factor, k_i. After some loop steps, the final values will be eliminated the total product of k_i that is known as a length factor K to give the output results.

In Adaptive CORDIC, instead of using the constant number of residual angles, the residual angles are variable and chosen by the previous state of z_i. It reduces the number of iterative steps but guarantees the equivalent results. Then, Adaptive CORDIC can decrease the design delay and resources.

Table 1. θ, c, and k in the Look-up Table with 16 constant angles.

i	θ	c	k
0	45.000000000	35.782525588	0.707106781
1	26.565051177	20.300647322	0.894427191
2	14.036243468	10.580629908	0.970142500
3	7.125016349	5.350675362	0.992277877
4	3.576334375	2.683122491	0.998052578
5	1.789910608	1.342542159	0.999512076
6	0.895173710	0.671393940	0.999877952
7	0.447614171	0.335712335	0.999969484
8	0.223810500	0.167858088	0.999992371
9	0.111905677	0.083929284	0.999998093
10	0.055952892	0.041964672	0.999999523
11	0.027976453	0.020982340	0.999999881
12	0.013988227	0.010491170	0.999999970
13	0.006994114	0.005245585	0.999999993
14	0.003497057	0.002622792	0.999999998
15	0.001748528	0.000874264	0.999999999

As an example, if we choose the input angle as $\boldsymbol{15^o}$ with 16 constant angles as Tab. 1, the result of rotations by conventional and Adaptive CORDIC is described in Eq. (6) and Eq. (7), respectively. The error of conventional method is 8.89E-04, while the proposed CORDIC has the error of 3.9E-04.

$$\boldsymbol{\theta_0 - \theta_1 - \theta_2 + \theta_3 + \theta_4 - \theta_5 + \theta_6 + \theta_7 + \theta_8 + \theta_9 + \theta_{10} - \theta_{11} - \theta_{12} - \theta_{13} + \theta_{14} + \theta_{15} = 15.000889212^o} \quad (6)$$

$$\boldsymbol{\theta_2 + \theta_6 + \theta_{10} + \theta_{12} - \theta_{15} = 14.999609769^o} \quad (7)$$

To choose the constant angle $\boldsymbol{\theta_i}$ in each iteration *i*, this method is based on the concept of parameters *c*, which are calculated by Eq. (8) and described in Tab. 1. Additionally, there is a factor $\boldsymbol{k_i}$ in each iteration *i*. With the constant number of iterations, the length factor *K* is constant, while the *K* is variable due to the different amount of iterations. Thus, the factor $\boldsymbol{k_i}$ of each rotation *i* is described in Tab. 1. The pseudo-code of Adaptive CORDIC operation is shown in Fig. 3. In each iteration *i*, a constant angle $\boldsymbol{\theta_i}$ is chosen in order that $\boldsymbol{z_i}$ converges on zero.

$$\boldsymbol{c_i} = \begin{cases} \frac{\boldsymbol{\theta_i + \theta_{i+1}}}{\boldsymbol{2}} & \boldsymbol{, 0 \le i \le (N-2)} \\ \frac{\boldsymbol{\theta_i}}{\boldsymbol{2}} & \boldsymbol{, i < 0 \; and \; i > (N-2)} \end{cases} \quad (8)$$

$i = j = 0;\ x_0 = 1; y_0 = 0; K = 1;$
while $|z_i| > c_{15}$ ***and*** $i < N$ ***do***
  if $|z_i| \in (c_{i+1}\,; c_i]$ ***then***
    $z_{j+1} = z_j - sign(z_j)\theta_i;$
    $x_{j+1} = x_j - sign(z_j) y_i 2^{-i};$
    $y_{j+1} = y_j + sign(z_j) x_i 2^{-i};$
    $K = K * k_i;\ j = j + 1$
  end if
  $i = i + 1;$
end while
$X = x_j * K; Y = y_j * K;$

Figure 3. The pseudo-code of Adaptive CORDIC.

3. DESIGN IMPLEMENTATION

3.1. Implementation of Radix-2 Multi-path Delay Commutator Fast Fourier Transform (Radix-2 MDC FFT)

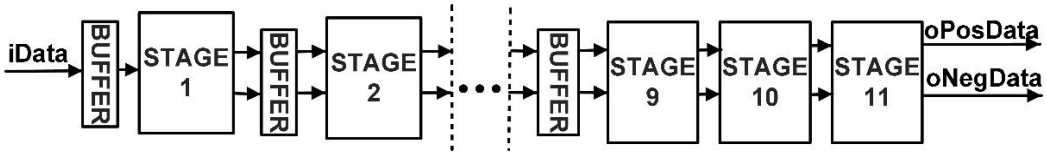


Figure 4. A block diagram of 2048-point Radix-2 MDC FFT.

In this study, the design of FFT with 2048-point is implemented. In according to Radix-2 MDC FFT architecture, with N-point design has $log_2(N)$ pipeline stages. As a result, this design is divided into 11 stages and its block diagram is shown in Fig. 4. During data transfer between

stages, operations in each module are not implemented in only one clock but several clocks. To guarantee data transfer, data buffers are used between every two stages. The block diagram of one stage in 2048-point Radix-2 MDC FFT is shown in Fig. 5.

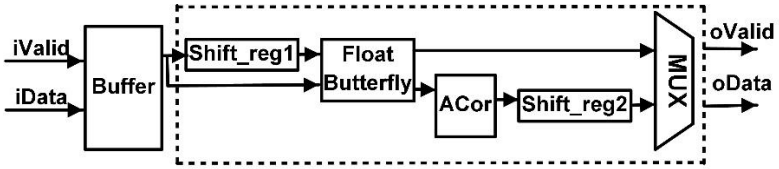


Figure 5. A block diagram of one stage in 2048-point Radix-2 FFT.

The operation of one stage is divided into four transactions as follows.

- Transaction 1:

  At the beginning of this transaction, the first $N/2$ data is stored in Shift_reg1. During that time, there are no output data. At each clock, control signals are always checked to determine the state of data whether transferred or hold.

- Transaction 2:

  The next $N/4$ data passes through the Buffer, combines with the delay data from Shift_reg1 and implements in floating-point Butterfly. The output result includes two parts, the adding (+) part will go directly to the next stage, while the subtracting (-) part will implement complex multiplication Adaptive CORDIC and the result will be stored in Shift_reg2.

- Transaction 3:

  The last $N/4$ calculates with the delay data from Shift_reg1 in floating-point Butterfly. The output result also consists of two parts, the adding (+) part continues to the next stage, while the subtracting (-) part will implement Adaptive CORDIC. The output data from Adaptive CORDIC is stored in Shift reg2 and puts the Shift_reg2 old data into the next stage.

- Transaction 4:

  Finally, the last $N/4$ data in Shift_reg2 in the third transaction is transferred to the next stage and completed this process.

In this design, there are three points of optimization including the first, the tenth and the last stages. These optimizations can reduce a lot of resources. At the first stage, due to the input data is the integer with 32 bits floating-point data format and only has the real value. The Adaptive CORDIC is optimized due that the imaginary part is 0 and ignored.

The optimization of the tenth stage is described in Fig. 6. As the Radix-2 architecture, the tenth stage only implements complex multiplication W_j. The W_j swaps and reverses the signs between real and imaginary values. Thus, it is designed by multiplexer and sign reverse.

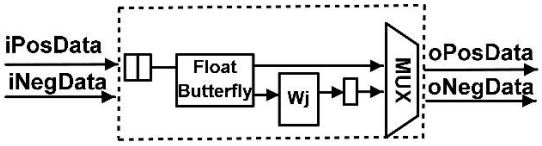


Figure 6. The optimization of the tenth stage.

The optimization of the last stage is shown in Fig. 7. The last stage does not have complex multiplication, only calculates the floating-point Butterfly and get the final output.

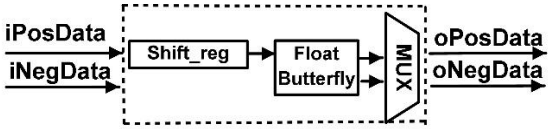


Figure 7. The optimization of the last stage.

3.2. Implementation of Adaptive CORDIC

The block diagram of Adaptive CORDIC is shown in Fig. 8 including four modules as rotation selection (RotSel), X and Y adder (FALU_XY), k_i multiplier (FMul_ki) and *K* multiplier, and output normalizer (FMulK_Norm).

The rotation selection (RotSel) module receives the input angle *iZ*, compares and chooses the relevant rotation angle. The output data consists of the sign and position of chosen angles. Besides, it informs if this angle is the last angle or not in the Tab. 1.

The X and Y adder module (FALU XY) receives the input data of *iX*, *iY* from the input data path of ACor, and the rotation angle from RotSel module. It implements the adder iteration and obtains the real and imaginary values of output as oY and oX, respectively. These values are in the floating-point data format of 1-bit sign, 8-bit exponent and 24-bit mantissa.

The k_i multiplier (FMul_ki) module is implemented at the same time with FALU_XY module. In each iteration, the KMul_ki module receives the information of rotation angle to choose the factor k_i corresponding to Tab.1 and calculates the product of these k_i. After that, the output is the length factor *oK*. Its data format has only 24-bit mantissa without the sign and exponent parts. The reason is that the length factor *K* is always positive, so the sign is always zero and ignored. Moreover, in the Tab.1, the higher *i*, the smaller *K* and *K* converges on one. Then, *K* cannot higher than 1 by multiplier iteration. As a result, 24-bit mantissa is sufficient to describe the length factor *K* and the sign and exponent parts can be omitted.

The *K* multiplier and output normalizer module (FMulK_Norm) receives data from the FALU_XY and FMul_ki modules. Then, the FMulK_Norm implements *K* elimination and normalizes the output data based on IEEE754 floating-point data format. The output will be in 32-bit floating-point with 1-bit sign, 8-bit exponent, and 23-bit mantissa.

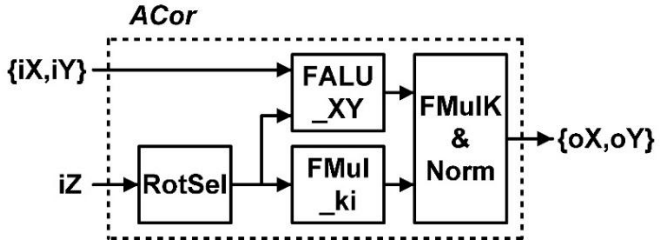


Figure 8. The block diagram of Adaptive CORDIC.

4. EXPERIMENTAL RESULTS

In this study, this design is built and implemented based on three kinds of Look-up Table (LUT) that cost 16, 8 and 4 constant angles θ. These LUTs are described in Tab. 1, Tab. 2, and

Tab. 3, respectively. The purpose of these implementations is to determine which is the most relevant number of constant angles to achieve the high accuracy but guarantee the optimizing speed.

The experimental results consist of two parts such as Adaptive CORDIC and Radix-2 MDC 2048-point FFT. Both are built and verified on Altera FPGA Stratix IV EP4SGX230KF40C2 chips with parameters of speed, resources and accuracy.

The resources are described by the number of logic elements (ALUTs) and registers. Regarding speed, the maximum frequency (Fmax), delay calculated by the number of clocks (Latency) and throughput. The throughput describes the performance of a design second. The accuracy is measured by comparing between the simulation result of hardware and the result of MATLAB, including Mean Square Error (MSE), and maximum error-ratio. While Mean Square Error is an essential parameter for a DSP system, the maximum error-ratio is also required besides the MSE value for floating-point implementation.

3.3. Experimental Results of Adaptive CORDIC

Table 2. θ, c, and k in the Look-up Table with 8 constant angles.

i	θ	c	K
0	45.000000000	35.782525588	0.707106781
1	26.565051177	20.300647322	0.894427191
2	14.036243468	10.580629908	0.970142500
3	7.125016349	5.350675362	0.992277877
4	3.576334375	2.683122491	0.998052578
5	1.789910608	1.342542159	0.999512076
6	0.895173710	0.671393940	0.999877952
7	0.447614171	0.335712335	0.999969484

Table 3. θ, c, and k in the Look-up Table with 4 constant angles.

i	θ	c	K
0	45.000000000	35.782525588	0.707106781
1	26.565051177	20.300647322	0.894427191
2	14.036243468	10.580629908	0.970142500
3	7.125016349	5.350675362	0.992277877

According to complex multiplication (W) characteristics in FFT algorithm, a 2048-point FFT design only needs 1024 angles from $0\pi/1024$ to $1023\pi/1024$. Thus, Adaptive CORDIC is measured with 1024 input data in each experiment. And the throughput shows how many samples are implemented in one second (Mega sample per second - MSps) and it is calculated based on Eq. (9). Regarding precision, Mean Square Error (MSE) and maximum error-ratio (part

per a million - ppm) calculated by Eq. (10), Eq. (11), and Eq. (12). Tab. 4 shows experimental results on Altera Stratix IV EP4SGX230KF40C2 chip of three complex multiplications based Adaptive CORDIC such as ACor_16, ACor_8, and ACor_4 corresponding to three kinds of Look-up Table: 16, 8, and 4 constant angles in Tab. 1, Tab. 2 and Tab. 3, respectively.

$$Throughput = \frac{Fmax \times 1024}{Latency} \tag{9}$$

$$MSE = \frac{1}{1024} \sum_{i=1}^{1024} SW\ i\ - HW\ i\ ^2 \tag{10}$$

$$Error - ratio = \frac{SW\ i\ - HW\ i}{SW\ i}\ ,\ i = 1,2,\dots,1024 \tag{11}$$

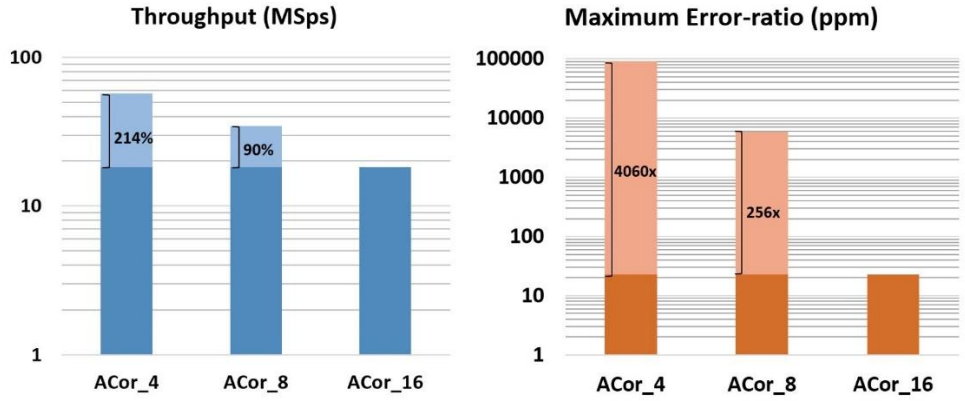
$$Maximum\ Error - ratio = \frac{MAX(Error - ratio)}{1000000} \tag{12}$$

where SW is the result of MATLAB, and HW is the output value of hardware.

Table 4. Experimental Results of Adaptive CORDIC on Altera FPGA Stratix IV chip.

Parameters	**ACor_16**	**ACor_8**	**ACor_4**
ALUTs	7,750 (4.25%)	7,219 (3.96%)	6,858 (3.76%)
Registers	625 (0.42%)	623 (0.416%)	604 (0.4%)
Fmax (MHz)	111.37	118.32	119.8
Latency (clocks)	6,621	3,497	2,145
Throughput (MSps)	18.215	34.647	57.191
Mean Square Error	1.103E-10	8.365E-06	2.028E-03
Maximum Error-ratio (ppm)	22.769	5851.998	92464.114

According to the experiment results of three Adaptive CORDIC designs in Tab. 4, the number of logic elements (ALUTs) of ACor_16, ACor_8 and ACor_4 are 4.25 %, 3.96 % and 3.76 % of total logic elements of Altera Stratix IV chip. Also, their registers are 0.42 %, 0.416 % and 0.4 % of the Altera Stratix IV chip's registers, respectively. Therefore, resources of three designs are roughly equivalent. However, it has a trade-off between the speed and accuracy and the comparisons are described in Fig. 9. Regarding speed, ACor_8 and ACor_4 have more computation times than ACor_16 that are implemented in one second by 90 % and 214 %, respectively. But concerning precision, maximum error-ratio of ACor_8 and ACor_4 are higher than maximum error-ratio of ACor_16 by 257 times and 4061 times, respectively. If the maximum error-ratio is higher, the accuracy of design is lower. Therefore, ACor_16 has the highest accuracy among three Adaptive CORDIC designs.



(a) (b)

Figure 9. The trade-off between the speed (a) and accuracy (b) in Adaptive CORDIC.

Beside this result on FPGA, these designs are synthesized on 65 nm SOTB technology. Silicon-On-Thin-Buried-Oxide (SOTB) CMOS is a SOI device for low power electronics due to its back-bias control and small variability [13]. The comparison between ACor_16, ACor_8, and ACor_4 on 65 nm SOTB technology, synthesized results after Placement and Routing step, are shown in Tab. 5. These designs have the same operation frequency at 100 MHz. There are some differences of area, logic cells and power consumption. The area of ACor_8 and ACor_4 are smaller than ACor_16 by 0.29 % and 5.1 %. The number of logic cells reduces when the number of constant angles in Look-up Table decreased. And the power consumption of ACor_16 and ACor_8 are lower than ACor_16 by 5.2 % and 9.23 %. These verifications of three designs on SOTB process are to prove that the decrease in the number of constant angles in Look-up Table effect directly on the accuracy of design much more than other factors.

Table 5. Results of Adaptive CORDIC on 65nm SOTB technology.

Parameters	ACor_16	ACor_8	ACor_4
Frequency (MHz)	100	100	100
Area (μm^2)	72,376	72,166	68,670
Logic cells	19,393	18,889	18,514
Power Consumption (μW)	672	637	610

3.4. Experimental Results of 2048-point Radix-2 MDC FFT

Tab. 6 shows experimental results on Altera Stratix IV EP4SGX230KF40C2 chip of three 2048-point Radix-2 MDC FFT designs based Adaptive CORDIC (FFT_ACor_16, FFT_ACor_8, and FFT_ACor_4) corresponding to three kinds of Look-up Table in Tab. 1, Tab. 2, and Tab. 3, respectively. The throughput shows how many 2048-point FFTs are implemented in one second (FFTs/s) and it is calculated based on Eq. (13). Regarding precision, Mean Square Error (MSE) and maximum error-ratio (part per a thousand - ppt) calculated by Eq. (14), Eq. (15) and Eq. (16).

$$Throughput = \frac{Fmax}{Latency} \tag{13}$$

$$MSE = \frac{1}{2048} \sum_{i=1}^{2048} SW\ i\ - HW\ i\ ^2 \tag{14}$$

$$Error - ratio = \frac{SW\ i\ - HW\ i}{SW\ i}\ ,\quad i = 1,2,\ldots,2048 \tag{15}$$

$$Maximum\ Error - ratio = \frac{MAX(Error - ratio)}{1000} \tag{16}$$

where SW is the result of MATLAB, and HW is the output value of hardware.

Table 6. Experimental Results of 2048-point Radix-2 MDC FFT on Altera FPGA Stratix IV chip.

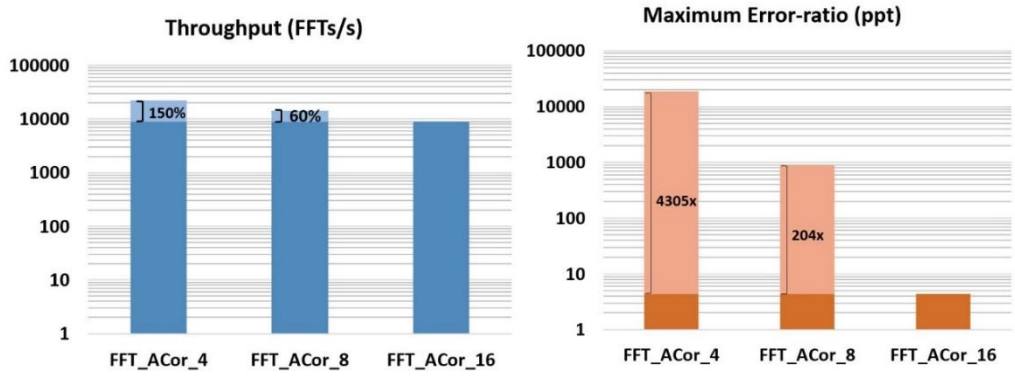
Parameters	FFT_ACor_16	FFT_ACor_8	FFT_ACor_4
ALUTs	91,760 (50.3%)	90,237 (49.5%)	89,725 (49.2%)
Registers	29,098 (16%)	29,207 (16%)	37,836 (21%)
Fmax (MHz)	109.33	107.07	112.84
Latency (clocks)	12,173	7,457	5,032
Throughput	8,981	14,358	22,424
Mean Square Error	6.206E-06	5.583E-01	2.043E+02
Maximum Error-ratio (ppm)	4.408	905.55	18,979

According to the experiment results of three 2048-point Radix-2 MDC FFT designs in Tab. 6, the number of logic elements (ALUTs) of FFT_ACor_16, FFT_ACor_8 and FFT_ACor_4 are 50.3 %, 49.5 % and 49.2 % of total logic elements of Altera Stratix IV chip. Also, their registers are 16 %, 16 % and 21 % of the Altera Stratix IV chip's registers, respectively. Because of increase in the throughput performance, so the bandwidth of data buffers between every two stages had to be enlarged. Thus, The FFT_ACor_4 consumed 21% of the total register on Stratix IV, and it's higher on 5% than the remaining ones. At last, the difference of resources in three 2048-point Radix-2 MDC FFT designs is not large. However, it has a trade-off between the speed and accuracy and the comparisons are described in Fig. 10.

With the speed performance, FFT_ACor_8 and FFT_ACor_4 computed the 2048-point FFTs than FFT_ACor_16 that are implemented in one second by 60 % and 150 %, respectively. But regarding precision, maximum error-ratio of FFT_ACor_8 and FFT_ACor_4 are higher than maximum error-ratio of ACor_16 by 204 times and 4305 times, respectively. As a result, FFT_ACor_16 has the highest accuracy performance.

Moreover, the experimental results on FPGA with the comparison with other designs is presented by the Tab. 7. All designs are implemented on Altera FPGA Stratix IV and the equivalent of point range. According to Tab. 7, the FFT_ACor_16 design costs more than [6]

design by 14.57 % logic elements, but its registers consume less than [6] by 1.62 times. Our proposed design has the precision of floating point, while the others are fixed point designs. Also, regarding speed, this proposal design has the roughly similar maximum frequency with [6], while it is faster than [14] design by 1.625 times. With the floating-point implementation, our proposed design still has the advantages of accuracy, and speed, and guarantees resource optimization.



(a) (b)

Figure 10. The trade-off between the speed (a) and accuracy (b) in 2048-point Radix-2 MDC FFT.

Table 7. FPGA implementation results in comparison with others.

Parameters	**FFT_ACor_16**	**[6]**	**[14]**
Chip	Stratix IV EP4SGX230KF40C2	Stratix IV EP4SGX530KH40C3	Stratix IV EP4SGX530KH40C3
Precision	Floating-point	Fixed-point	Fixed-point
Point range	2048	256 - 2048	128 - 2048
ALUTs	91,760	80,088	N/A
Registers	29,098	47,129	N/A
Frequency (MHz)	109.33	111	67.28

4. CONCLUSIONS

In this study, we present a design of High-precision High-accuracy 2048-point FFT using Single-precision Floating-point Adaptive CORDIC. This study includes two parts as Adaptive CORDIC design which stands for a complex multiplication in FFT system and a 2048-point Radix-2 Multi-path Delay Commutator FFT. These designs are built and verified with three kinds of Look-up Table that cost 16, 8, and 4 angles. As experimental results, there is an equivalence of resources but it has a trade-off between speed performance and accuracy. In conclusion, we draw that designs using the Look-up Table with 16 constant angles responding to resource and speed optimization, and a high precision of computations. Besides this, Adaptive CORDIC can be developed and optimized for various points of Fast Fourier Transform designs or other signal processing systems as future works.

Acknowledgement. This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under the grant number of B2017-18-05. This research is also in the cooperation result with Pham's laboratory at the University of Electro-Communications (UEC), Tokyo, Japan.

REFERENCES

1. Cooley J. W. and Tukey J. W. - An Algorithm for the Machine Calculation of Complex Fourier Series, Mathematics of Computation **19** (1965) 297-301.
2. Tang Y. and Vucetic B. - The FFT-based Multiuser Detection for DS-CDMA Ultra - wideband Communication Systems, Ultra Wideband Systems Joint with Conference on Ultra wideband Systems and Technologies, Kyoto, Japan (2004) 111-115.
3. Molisch A. F. and Zhang X. - FFT-based Hybrid Antenna Selection Schemes for Spatially Correlated MIMO Channels, IEEE Communications Letters **8** (1) (2004) 36-38.
4. Andrews J. G., Ghosh A., and Muhamed R. - Fundamentals of WiMAX: Understanding Broadband Wireless Networking, Prentice Hall Communications Engineering and Emerging Technologies Series, 2007.
5. Dahlman E. - 3G Evolution: HSPA and LTE for Mobile Broadband, Academic Press, 2008.
6. Dinh P. T. K., Lanante L., Nguyen M. D., Kurosaki M.., Ochi H. - An Area-Efficient Multimode FFT Circuit for IEEE 802.11 ax WLAN Devices, The 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 735-739.
7. Le C., Chan S., Cheng F., Fang W., Frischman M., Hensley S., Johnson R., Jourdan M., Marina M., Parham B., Rogez F., Rosen P., Shah B., and Taft S. - Onboard FPGA-based SAR Processing for Future Spaceborne Systems, Proceedings of the 2004 IEEE Radar Conference, Philadelphia, USA, 2004, pp. 15-20.
8. Li J., Saruni M. V., and Shannon L. - Scalable, High Performance Fourier Domain Optical Coherence Tomography: Why FPGAs and not GPGPUs, IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Salt Lake City, Utah, 2011, pp. 49-56.
9. Oruklu E., Xiao X., and Saniie J. - Reduced Memory and Low Power Architecture for CORDIC-based FFT Processors, Journal of Signal Processing Systems **2** (66) (2011) 129-134.
10. Radhouane R., Liu P., and Modlin C. - Minimizing the Memory Requirement for Continuous Flow FFT Implementation: Continuous Flow Mixed Mode FFT (CFMM-FFT), in IEEE International Symposium on Circuits and Systems (ISCAS 2000), Genève, Switzerland, 2000, pp. I-116–I-119.
11. Nguyen H. T., Nguyen X. T., Hoang T. T., Le D. H., and Pham C. K. - A Low-resource Low-latency Hybrid Adaptive CORDIC with Floating-point Precision, IEICE Electronics Express (ELEX) **12** (9) (2015) 1-12.
12. Vo T. P. T. , Hoang T. T., Pham C. K., and Le D. H. - A Floating-point FFT Twiddle Factor Implementation Based on Adaptive Angle Recoding CORDIC, in The 2017 IEEE Int. Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom), Danang, Vietnam, 2017, pp. 21-26.

13. Kamohara S., Sugii N., Yamamoto Y., Makiyama H., Yamashita T., Hasegawa T., Okanishi S., Yanagita H., Kadoshima M., Maekawa K., Mitani H., Yamagata Y., Oda H., Yamaguchi Y., Ishibashi K., Amano H., Usami K., Kobayashi K., Mizutani T., Hiramoto T.- Ultralow-voltage design and technology of silicon-on-thin-buried-oxide (SOTB) CMOS for highly energy efficient electronics in IoT area, Symposium on VLSI Technology (VLSI-Technology): Digest of Technology Papers, Honolulu, 2014.
14. Adiono T., and Mareta R. - Low Latency Parallel-Pipelined Configurable FFT-IFFT 128/256/1024/2048 for LTE, 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012) **2** (2012) 768-773.