On a Wideband Fast Fourier Transform for a Radio Telescope

Hiroki Nakahara[*], Hiroyuki Nakanishi[†], Tsutomu Sasao[‡]
[*†]Kagoshima University, Japan, [‡]Kyushu Institute of Technology, Japan

ABSTRACT

The radio telescope analyzes a radio frequency from celestial objects by using fast Fourier transform (FFT). In this application, its bandwidth f is wider than that of the typical FFT. Since the amount of hardware for the typical FFT circuit is proportional to the bandwidth f, a special technique is necessary for this application. This paper shows a realization of wideband FFT for the radio telescope on an FPGA. We show that the memory size for the conventional FFT, which consists of the twiddle factor memory and the transpose memory, is too large. We replace the twiddle factor memory with the pipelined CORDIC. To reduce the number of transpose memories, we increase the radix of the FFT from 2^2 to 2^k, also we use the DDR2SDRAM to implement the transpose memory. We implement the 2^{30}-FFT on an Altera's Stratix IV GX530 FPGA. It performs the 2^{30}-FFT operations in 1.5 seconds. Compared with the Altera's FFT library, our FFT circuit realizes 2^{14} times wider bandwidth on the same FPGA. Also, compared with Tesla S1070 utilizing four GPUs, our FFT circuit is faster and dissipates lower power.

1. INTRODUCTION

1.1 Radio Astronomy and Radio Telescope

Radio Astronomy studies celestial objects at the radio frequency (RF). In 1931, Karl Jansky observed the RF (14.6 meter wave length) coming from the Milky Way. Since then, observation of the RF by the radio telescope found that celestial objects take a different stance in the RF unlike in the visible light. The development in radio astronomical observatory is expected to help us learn more about the newborn Galaxy in early universe, a birth of a star, an evolution of materials in space, and so on.

Nowadays, the SKA (Square Kilometer Array) develops a next generation radio telescope with a large collecting area (a square kilometer) for the RF [19]. Fig. 1 shows a typical radio telescope. First, the antenna receives the RF coming from the celestial objects. Then, the feedhone sends the electromagnetic wave to the receiver through the waveguide. Next, the receiver transforms it to the intermediate frequency (IF) by using the amplifier and the mixer. Finally, **the spectrometer** converts the voltage wave to the power spectrum.

The early spectrometer uses the filter bank consisting of many analog filters for the specified frequency. However, in the filter bank spectrometer, the bandwidth is too narrow. Next, the acousto-optical spectrometer (AOS) based on the diffraction of light at ultrasonic waves has been developed. However, in the AOS, it is difficult to adjust non-linear characteristics in the analog device (such as piezo element). Nowadays, digital spectrometers are used for a wide range of radio telescopes. Fig. 1 also shows the digital spectrometer.

[*]e-mail:nakahara@eee.kagoshima-u.ac.jp
[†]e-mail:hnakanis@sci.kagoshima-u.ac.jp
[‡]e-mail:sasao@cse.kyutech.ac.jp

This work was presented in part at the third international workshop on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART2012), Naha, Okinawa, Japan, May 31-June 1, 2012.

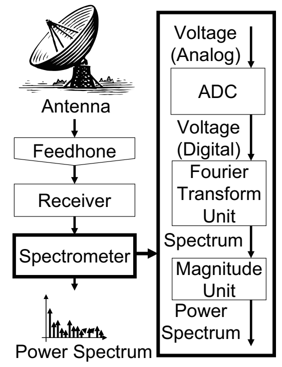

Figure 1: Ratio telescope.

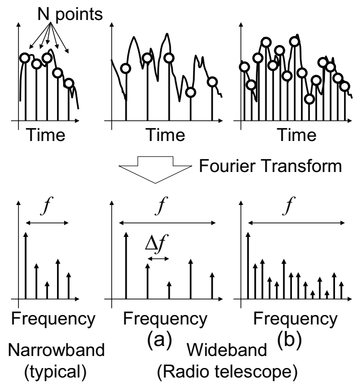

Figure 2: Problem in wideband analysis.

First, the analog-to-digital converter (ADC) converts the IF voltage wave to the digital signal. Then, the Fourier transform unit performs **the fast Fourier transform (FFT)** to obtain the spectrum. Finally, the magnitude unit calculates the power spectrum. The computation of magnitude is a light task, while that of Fourier transform is a heavy task. Thus, in the digital spectrometer, the Fourier transform hardware is needed.

1.2 Requirement and Problem of Wideband Analysis

Recently, benefits for the wideband analysis of the RF have been reported [8]. The first benefit is the expansion of measurement range of physical parameter by obtaining the wideband spectrum. For example, by obtaining spectrum for atoms, bright lines from ionized gas, we can investigate new gas component and measurement of temperature between the stars. The second benefit is the improvement of sensitivity of the radio telescope. The noise of the measurement data is proportional to $\frac{1}{\sqrt{f}}$, where f is the bandwidth. Thus, the increase of f enhances the sensitivity. As a result, we can detect more dark stars.

Given the fixed number of points N, wider bandwidth decreases the resolution Δf (Fig. 2 (a)). This causes measurement errors. For example, the SERENDIP (Search for Extraterrestrial Radio Emissions from Nearby Developed Intelligent Populations), which is one of a SETI (Search for Extra-Terrestrial Intelligence) project [18], uses **the SETI spectrometer** analyzing $f = 200$ MHz bandwidth with $\Delta f = 2$ Hz in one second [16]. Thus, it requires the 2^{27}-point FFT operation. In other instance, to detect Zeeman effect ($\Delta f = 1963.27$ Hz) in the OH emission line ($f = 1.66$ GHz) for interstellar gas, it requires at least the 2^{20}-point FFT operation. Therefore, in the radio telescope, a large N is required (Fig. 2 (b)). Table 1 compares N for the radio telescope with that for commercial purpose FFTs. The FPGA implementation of the conventional FFT requires $O(log_2 N)$ multipliers and $O(N log_2 N)$ bits of memory [23]. Table 1 implies that, for the radio telescope, the memory for the conventional FFT would be too large to implement.

Table 1: Comparison of the number of points N.

	Application	N
Commercial Purpose [22]	Baseband 3GPP	$2^{10} - 2^{11}$
	OFDM	2^8
	CT scanner	$2^{10} - 2^{13}$
	High Quality Sound	2^9
Radio Telescope	SETI Spectrometer	2^{27}
	Zeeman Effect Observation	2^{20}

1.3 Limitation for Radio Telescope

Since the RF from celestial objects is absorbed by water molecules and oxygen molecules, the radio telescope is located at the altitude of 4,000-5,000 meters summit of mountains[1]. Also, since the RF with long wavelength is reflected at ionization layer, a satellite measurement at outer space is necessary. In 1997, the radio telescope satellite HALCA (Highly Advanced Laboratory for Communications and Astronomy) was on a mission [6]. Therefore, in the radio telescope, the following location conditions are desirable:
Limited Space: Since the ratio telescope is located at the summit of mountains or outer space, the super computer requiring huge space is unacceptable. High-speed communication wires cannot set in a narrow space. Therefore, compact hardware is desirable.
Low-Power Consumption: At the summit of mountains or outer space, the power source is also limited. Thus, low-power consumption is required. For example, NASA's MARVEL (Mars Volcanic Emission and Life) mission requires a circuit that dissipates less than 10 W power [15]. Although the GPU outperforms the FPGA [10], it dissipates much higher power than the FPGA.

1.4 Related Works

Cooley and Tukey proposed **the Fast Fourier Transform (FFT)** based on the radix two [3]. Duhamel *et al.* proposed the split-radix FFT [4]. Radix three, 2^2, four, five, and eight FFTs were proposed in [14, 11, 5]. As for the FFT implementation, He *et al.* proposed the radix 2^2 single delay path feedback FFT [7]. Knight and Rabiner analyzed error with respect to the fixed point FFT [13, 17]. Zhou *et al.* analyzed the amount of hardware [23]. Andraka replaced the twiddle factor memory with the pipelined CORDIC [1]. Kondo *et al.* implemented the equivalent SETI spectrometer on the NVIDIA's Tesla S1070 (four GPUs) [12].

1.5 Proposed Method

Because of restrictions for the space and the power consumption, we adopt a single-FPGA board for the wideband FFT. In the FFT, we need the twiddle factor memory and the transform memory. First, to remove the twiddle factor memory, we replace it with the pipelined CORDIC, since the pipelined CORDIC requires no memory. Then, we extend the radix of the FFT from 2^2 to 2^k in order to reduce the number of transform memories. We realize the transform memory by the DDR2SDRAM. In this way, we implemented the wideband FFT. Contributions of this paper are as follows:
We implement the 2^{30}-point FFT on an FPGA board, and obtained its performance and the necessary amount of hardware: As far as we know, this paper first reports the implementation of the 2^{30}-point FFT on an FPGA board. The conventional FFT library [2] can implement up to 2^{16}-point FFTs.
We show that the FPGA based-one is faster and dissipates lower power than the GPU based-one: As for high productivity computing, GPUs outperform the FPGAs [10]. However, the radio telescope has a special requirements; not the productivity but the small space and the low-power consumption.

[1]Currently, ALMA (Atacama Large Millimeter/submeter Array) project is going to build a new radio telescope at the Andes Mountains.

The rest of the paper is organized as follows: Chapter 2 shows the bottleneck of the conventional FFT on an FPGA; Chapter 3 replaces the twiddle factor memory with the pipelined CORDIC; Chapter 4 extends the radix of the FFT from 2^2 to 2^k; Chapter 5 shows the experimental results; and Chapter 6 concludes the paper.

2. FAST FOURIER TRANSFORM (FFT)

First, we explain the N-point discrete Fourier transform, then, we introduce the N-point fast Fourier transform. Next, we analyze the amount of hardware for the radix 2^2 FFT.

2.1 N-Point Discrete Fourier Transform

Let $x(n), (n = 0, 1, \ldots, N-1)$ be the input signal. **The N-point discrete Fourier transform (N-DFT)** is

$$X(m) \quad = \quad \sum_{n=0}^{N-1} x(n) W_N^{nm}, (m = 0, 1, \ldots, N-1), \quad (1)$$

where W_N^{nm} is **a twiddle factor**. It is defined as

$$W_N^{nm} \quad = \quad e^{-j\frac{2\pi}{N}nm} = \cos(\frac{2\pi}{N}nm) + j\sin(\frac{2\pi}{N}nm), \quad (2)$$

where j is an imaginary unit. In this paper, we assume that the input signal $x(n)$ is a complex number, and N is a power of two.

2.2 N-Point Fast Fourier Transform

Since the direct implementation for Expr. (1) requires $O(N^2)$ multipliers, it is impractical for large N. Cooley and Tukey proposed **the N-point fast Fourier transform (N-FFT)** that reduces the number of multipliers to $O(Nlog_2N)$. In this paper, we adopt **the decimation-in-time N-FFT**.

Here, we convert the N-DFT into the N-FFT. In Expr. (1), suppose that N input signals are decomposed into odd indices and even ones, then we have

$$X(m) \quad = \quad \sum_{n=0}^{N/2-1} x(2n) W_N^{2nm} + \sum_{n=0}^{N/2-1} x(2n+1) W_N^{(2n+1)m}.$$

Since $W_N^{2mn} = e^{-j(2\pi/N)2mn} = e^{-j(2\pi/(N/2))mn} = W_{N/2}^{mn}$, we have

$$\begin{aligned} &= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nm} + W_N^m \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nm} \quad (3) \\ &= DFT_{even}(m) + W_N^m DFT_{odd}(m), \end{aligned}$$

where $m = 0, 1, \ldots, \frac{N}{2}-1$. Expr. (3) means that the N-DFT is decomposed into two $\frac{N}{2}$-DFTs. By applying Expr. (3) to the N-DFT until decomposed into 1-DFTs, we have the N-FFT. In this case, we apply Expr. (3) $s = log_2N$ times recursively. We define s as **the number of stages**.

The butterfly operator shown in Fig. 3 performs a basic operation in N-FFT. The butterfly operator consists of a complex adder (subtracter) (Fig. 4 (a)) and a complex multiplier (Fig. 4 (b)). The butterfly operator shown in Fig. 3 (a) uses two multipliers. By using symmetric property $W_N^{m+N/2} = -W_N^m$, we can share the multiplier as shown in Fig. 3 (b).

EXAMPLE 2.1. *Fig.5 shows a dataflow for the 8-FFT. In this case, the number of stages s is $log_2 8 = 3$. Note that, we use butterfly operators shown in Fig. 3 (b).* ■

Each stage requires $\frac{N}{2}$ multipliers. Thus, the N-FFT consisting of s stages requires $O(Nlog_2N)$ multipliers. N-DFT requires $O(N^2)$ multipliers, so N-FFT reduces the number of multipliers.

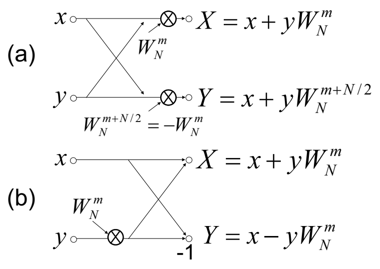

Figure 3: Butterfly operator.

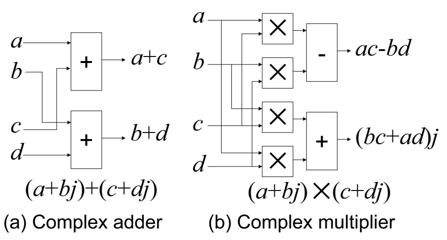

Figure 4: Complex operators.

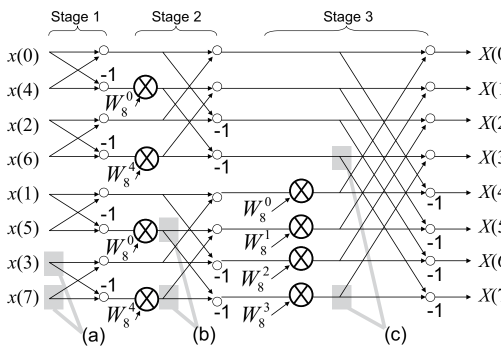

Figure 5: Dataflow for the 8-FFT.

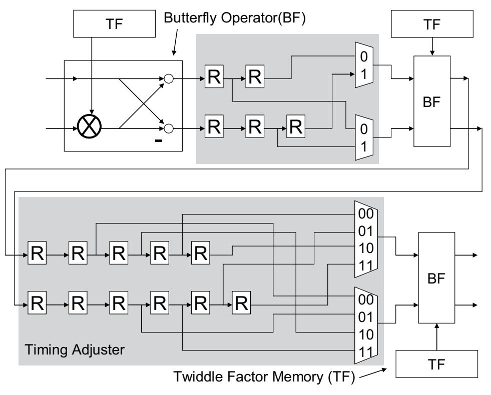

Figure 6: radix-2^3 butterfly unit for the 8-FFT.

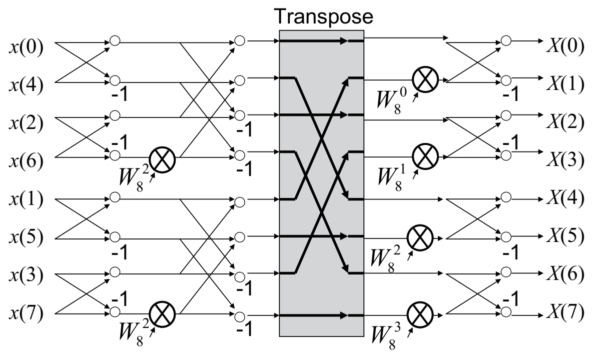

Figure 7: Example of transpose.

2.3 Radix-2^k Butterfly Unit

The butterfly operator can be realized by adders and multipliers on an FPGA. The twiddle factor is stored into **the twiddle factor memory**. We assign the stage numbers as $1, 2, \ldots, log_2N$ from the input to the output. In the stage 1, the inputs for the butterfly operation are applied to the adjacent points, then, the results are sent to the adjacent points (Fig. 5 (a)). In the stage 2, the inputs for the butterfly operation are applied to the distance-two points, then, the results are sent to distance-two points (Fig. 5 (b)). In the stage 3, the inputs for the butterfly operation are applied to the distance-four points, then, the results are sent to distance-four points (Fig. 5 (c)). Thus, **the timing adjuster** is necessary between adjacent stages. **The radix-2^k butterfly unit** for the N-FFT can be realized by the timing adjuster, the butterfly operator, and the twiddle factor memory [7].

EXAMPLE 2.2. *Fig. 6 shows the radix-2^3 butterfly unit realizing the 8-FFT shown in Fig. 5.* ■

Here, we analyze the amount of hardware for the radix-2^k butterfly unit. Let $s = \lceil log_2N \rceil$ be the number of stages for the N-FFT. The number of multipliers[2] Mul_{R2^k} is $O(log_2N)$, since

$$Mul_{R2^k} = 4s = 4\lceil log_2N \rceil. \quad (4)$$

Two adders are necessary for the complex multiplier, and four adders are necessary for two complex adders. Thus, the number of adders Add_{R2^k} is $O(log_2N)$, since

$$Add_{R2^k} = (4+2)s. \quad (5)$$

For stage i, the number of necessary registers is $2^i + 2(i-1) - 1$. Note that, the stage 1 use no register. Thus, the number of registers Reg_{R2^k} is $O(N)$, since

$$Reg_{R2^k} = \sum_{i=2}^{s}(2^i + 2(i-1) - 1). \quad (6)$$

We assume that the selector consists of multiplexors with two data inputs (2-MUXs). For stage i, a pair of selectors consisting of $2^{i-1} - 1$ copies of 2-MUXs are necessary. Thus, the number of 2-MUXs is $O(N)$, since

$$Mux_{R2^k} = 2 \times \sum_{i=2}^{s}(2^{i-1} - 1). \quad (7)$$

The amount of the twiddle factor memory $TwiddleMem_{R2^k}$ is $O(Nlog_2N)$, since

$$TwiddleMem_{R2^k} = Ns. \quad (8)$$

Therefore, for the radix-2^k butterfly unit, registers, 2-MUXs, and twiddle factor memories will be the bottleneck for the implementation.

2.4 Radix-2^2 Butterfly Units with Transpose

For radix-2^k butterfly units, since different stages handle points with different distances, numbers of registers and 2-MUXs increase. By applying **the transpose operation** replacing indices between stages, we can adjust the points of the inputs for the butterfly operator. As a result, numbers of registers and 2-MUXs are reduced. **The transpose memory** performs the transpose operation.

EXAMPLE 2.3. *Fig. 7 shows an example of applying transpose operation between stage 2 and stage 3 on the dataflow shown in Fig. 5. In this case, since the inputs for the butterfly operators become adjacent points, no timing adjuster is necessary.* ■

Previous work showed that, for small N, **the radix-2^2 butterfly units with transpose ($R2^2FFT$)** shown in Fig. 8 becomes small. Since the $R2^2FFT$ uses $s = log_2N$

[2]We implement the multiplication by the multiplier on the FPGA. Thus, we assume that the number of multiplications is equal to that of multipliers.

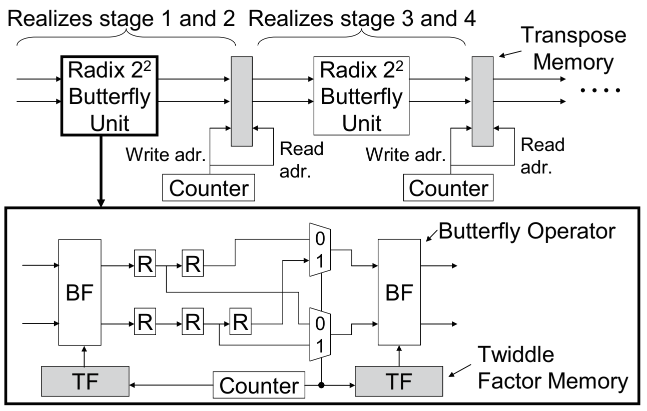

Figure 8: Radix-2^2 butterfly units with transpose ($R2^2FFT$).

butterfly operators, the number of multipliers Mul_{R2^2FFT} is equal to Expr. (4), and the number of adders Add_{R2^2FFT} is equal to Expr. (5). Also, it uses $\frac{s}{2}$ radix-2^2 butterfly units. Thus, the number of registers Reg_{R2^2FFT} is $O(log_2N)$, since

$$Reg_{R2^2FFT} \quad = \quad 5s. \tag{9}$$

The number of 2-MUXs Mux_{R2^2FFT} is $O(log_2N)$, since

$$Mux_{R2^2FFT} \quad = \quad 2s. \tag{10}$$

The amount of twiddle factor memory $TwiddleMem_{R2^2FFT}$ is equal to Expr. (8). Thus, it increases with $O(Nlog_2N)$. In the $R2^2FFT$, $\frac{s}{2}$ transpose memories with N bits are used. Thus, the amount of transpose memory $TranMem_{R2^2FFT}$ is $O(Nlog_2N)$, since

$$TranMem_{R2^2FFT} \quad = \quad N\frac{s}{2}. \tag{11}$$

For large N ($> 2^{20}$), both of twiddle factor and transpose memories with $O(Nlog_2N)$ bits are too large to implement. Therefore, we reduced these memories.

3. REPLACEMENT OF TWIDDLE FACTOR MEMORY WITH PIPELINED CORDIC

From Expr. (8), for the $R2^2FFT$, the twiddle factor memory is too large for large N. Expr. (2) shows that the twiddle factor is specified by the trigonometric functions (sin and cos). Thus, the twiddle factor memory can be replaced with the arithmetic circuit. Volder *et al.* proposed **COordinate Rotation DIgital Computer (CORDIC)** algorithm [21] that calculates the trigonometric function by a repetition of addition and shift operations. The FFT circuit for the CORDIC algorithm has been proposed [1]. The following is the CORDIC algorithm to calculate sin(z) and cos(z) for given radian z.

ALGORITHM 3.1. *Let N be an integer (a large N corresponds to a high precision), z be a given radian, x be the horizontal variable, y be the vertical variable, sign and tmp be temporary variables.*

```
Step 1 $r_0 \leftarrow \sqrt{2}$, $\theta[i] = \arctan(\frac{1}{2i}), (i = 0, 1, \ldots, log_2N)$,
       $r_i \leftarrow \frac{r_{i-1}}{cos(\theta[i])}, (i = 1, 2, \ldots, log_2N)$.
Step 2 for $i \leftarrow 0$ until $i < log_2N$ begin
    Step 2.1  if $(z > 0)$ then $sign \leftarrow -1$ else $sign \leftarrow 1$.
    Step 2.2  $z \leftarrow z + sign \times \theta[i]$.
    Step 2.3  $tmp \leftarrow x$.
    Step 2.4  $x \leftarrow x + z\frac{y}{2^i}$.
    Step 2.5  $y \leftarrow y + z\frac{tmp}{2^i}$.
    Step 2.6  $i \leftarrow i + 1$.
  end
Step 3 $\sin(z) \leftarrow \frac{y}{r_i}$, $\cos(z) \leftarrow \frac{x}{r_i}$.
Step 4 Terminate.
```

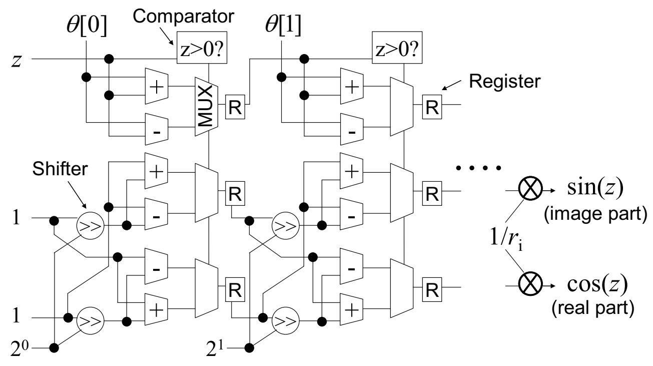

Figure 9: Pipelined CORDIC.

To calculate sin(z) and cos(z), Algorithm 3.1 performs the addition and the shift operations (Step 2.1-2.6) by using pre-computed values (Step 1). However, the direct implementation of Algorithm 3.1 tends to be slow, since it requires $\lceil log_2N \rceil$ repetitions of addition and shift operations (Step 2). The repetition can be converted to **the pipelined CORDIC** shown in Fig. 9. Since the pipelined CORDIC requires no memory, we can remove the twiddle factor memory. Although the pipelined CORDIC requires additional hardware to the original CORDIC, its throughput is higher than the original one.

Let $p = \lceil log_2N \rceil$ be the number of pipeline stages. The number of multipliers Mul_{COR} is $O(1)$, since

$$Mul_{COR} \quad = \quad 2. \tag{12}$$

The number of adders Add_{COR} is $O(log_2N)$, since

$$Add_{COR} \quad = \quad 6p. \tag{13}$$

For each pipeline stage, registers retain the horizontal value x, the vertical value y, and the rectangle z. Thus, the number of registers Reg_{COR} is $O(log_2N)$, since

$$Reg_{COR} \quad = \quad 3p. \tag{14}$$

Suppose that the selector and the comparator consist of 2-MUXs. The number of 2-MUXs Mux_{COR} is $O(log_2N)$, since

$$Mux_{COR} \quad = \quad 4p. \tag{15}$$

Although the $R2^2FFT$ uses log_2N pipelined CORDICs, its hardware is feasible.

4. REDUCTION OF TRANSPOSE MEMORY BY RADIX-2^K FFT

4.1 Radix-2^k Butterfly Units with Transpose

Expr. (11) shows that, the $R2^2FFT$ requires transpose memories with $O(Nlog_2N)$ bits. As for the radio telescope, the target number of points N is greater than 2^{27}. The DDR2SDRAM (Double-Data-Rate2 Synchronous Dynamic Random Access Memory)[3] with more than 2^{30} bits can realize the transpose memory. However, for the FPGA, the number of high-speed connectors for the DDR2SDRAM is limited [4]. For the radio telescope, since the high-speed ADC and the PCI express use the high-speed connectors, the off-chip DDR2SDRAM cannot use all the high-speed connectors. Thus, the number of transpose memories becomes a bottleneck.

[3]In the implementation, we used PC2-5300 DDR2SDRAM (dual channel), its bandwidth is 10.666 Giga Bytes per second.

[4]For example, Altera's Stratix IV GX530 has at most 12 high-speed connectors [2].

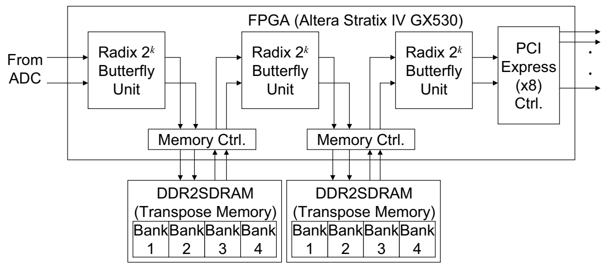
FPGA (Altera Stratix IV GX530)
From ADC
Radix 2^k Butterfly Unit
Radix 2^k Butterfly Unit
Radix 2^k Butterfly Unit
PCI Express (x8) Ctrl.
To PC
Memory Ctrl.
Memory Ctrl.
DDR2SDRAM (Transpose Memory)
Bank 1 Bank 2 Bank 3 Bank 4
DDR2SDRAM (Transpose Memory)
Bank 1 Bank 2 Bank 3 Bank 4

Figure 10: Radix-2^k butterfly units with transpose (R2^kFFT).

To reduce the number of transpose memories, we use **the radix-2^k butterfly units with transpose (R2^kFFT)** shown in Fig. 10. In the R2^kFFT, the transpose memory is inserted per k stages, and the pipelined CORDIC calculates the twiddle factor. Let $s = \lceil log_2 N \rceil$ be the number of stages for N-FFT. The number of transpose memories for the R2^kFFT is $\lceil \frac{s}{k} \rceil - 1$. Thus, the increase of k decreases the number of transpose memories. We use the off-chip DDR2SDRAM to realize transpose memories. The DDR2SDRAM operates at high clock frequency for double data rate, and reads and writes operations continuously by using multi-bank access and posted CAS operations. Thus, it performs high-speed transpose operations. In our implementation, we do not use the transpose memory at the output of the circuit, since the host PC converts the order of the index of the output. Also, at the input, the ADC board [9] converts the order of the input by using the high-speed buffer memory.

4.2 Analysis of Amount of Hardware

Consider an N-FFT with $s = \lceil log_2 N \rceil$ stages. The R2^kFFT with transpose has $q = \frac{s}{k}$ radix-2^k butterfly units. The number of pipeline stages for the pipelined CORDIC is $p = \lceil log_2 N \rceil$. From Exprs. (4) and (12), the number of multipliers Mul_{R2^kFFT} is $O(log_2 N)$, since

$$Mul_{R2^kFFT} = 4s + 2p = 6log_2 N.$$

From Exprs. (5) and (13), the number of adders Add_{R2^kFFT} is $O((log_2 N)^2)$, since

$$Add_{R2^kFFT} = 6s + 6ps = 6log_2 N(1 + log_2 N).$$

From Exprs. (6) and (14), the number of registers Reg_{R2^kFFT} is $O(\frac{2^k}{k} log_2 N)$, since

$$Reg_{R2^kFFT} = q \sum_{i=2}^{k} (2^i + 2i - 1) + 3ps. \qquad (16)$$

From Exprs. (7) and (15), the number of 2-MUXs Mux_{R2^kFFT} used for the selector is $O(\frac{2^k}{k} log_2 N)$, since

$$Mux_{R2^kFFT} = q \sum_{i=2}^{k} (2^i - 1) + 4ps. \qquad (17)$$

From Expr. (11), the amount of transpose memory is $O(Ns) = O(Nlog_2 N)$. With large k, we assume that s becomes a constant. Thus, $TransMem_{R2^kFFT}$ is $O(N)$. For the given number of stages s, Exprs. (16) and (17) show that, an increase of k increases numbers of registers and 2-MUXs. Thus, we must find the suitable k that is feasible for given hardware. In our implementation, k is obtained experimentally.

5. EXPERIMENTAL RESULTS

5.1 Implementation Environment

We implemented the R2^kFFT on the Altera's DE4 development and education board (FPGA: Altera's Stratix IV GX530, ALUTs 424,960, REGs: 424,960, M9ks: 1,280, and 18×18 DSP blocks: 1,024). This board has two DDR2SO-DIMMs, one PCI Express (×8), and two HSMCs (high-speed connector) to connect the ADC. We used the Altera's Quartus II version 11.0 to synthesis, and the Altera's MegaCore Function to generate the DDR2SDRAM controller and the PCI Express controller.

5.2 Selection of Optimal k

Exprs. (16) and (17) show that, numbers of registers and 2-MUXs increase with $O(q2^k)$. The DE4 FPGA board has two DDR2SO-DIMMs. We implemented the R2^kFFT consisting of three radix-2^k butterfly units as shown in Fig. 10, where $k = \lceil \frac{log_2 N}{3} \rceil$.

5.3 Implementation of 2^{30}-FFT

We implemented the 2^{30}-FFT using 189,948 ALUTs and 120 DSPs. The maximum clock frequency was 380.2 MHz. To run the DDR2SO-DIMM at 667 MHz, we set the system clock frequency to 333 MHz by using the PLL. Since the implemented FFT handles two points per one clock (333 MHz), it handles per $\frac{1}{333 \times 2^{20}}$ second. Thus, the operation time to handles 2^{30} points was $\frac{2^{30}}{333 \times 2^{20}} \times \frac{1}{2} = 1.537$ second. We used the ADC1x5000-8 (dual 2.5 Giga sample per second, 4bit). The implemented FFT extended to 36-bit fixed point two's complement complex numbers (18-bit real and 18-bit imaginary). In the butterfly operation in each stage of the FFT, the width grows by 1 bit by the adder and subtracter. This may yield $log_2 N + w$ bit outputs for w-bit data. To avoid an overflow, the scaling schedule is used to divide the values by a factor of $2^{\lceil (4+N-w/k) \rceil}$ in each $R2^k$ butterfly operator. Although this increases the scaling error, the radio astronomy requires the high-speed and wide band operation rather than the precision. To reduce error in the FFT operations, we used the rounding and saturation unit (RSU) in each DSP block. The SETI spectrometer also uses 36-bit fixed point two's complement complex numbers in the FFT part. To reduce hardware of the FFT part, the SETI spectrometer partitions the frequency band to r bands by using FIR filters, and realizes smaller $\frac{N}{r}$-FFTs [16]. On the other hand, the implemented FFT directly performs the FFT operation. Thus, the implemented FFT has a smaller error margin than the SETI spectrometer.

5.4 Comparison with Altera's FFT Library

We compared the implemented FFT with the Altera's MegaCore Function FFT library adopting the $R2^2FFT$ with respect to the amount of hardware (M9ks, ALUTs, and DSPs) and the maximum frequency. Fig. 11 (a) compares of the number of M9ks. Since the Altera's FFT library consumes 93.7% of available M9ks for $N = 2^{16}$, it cannot realize the 2^N-FFT with $N > 16$. On the other hand, the implemented FFT consumes no M9k. Fig. 11 (b) compares of the numbers of DSPs. Since the implemented FFT uses the pipelined CORDIC, it consumes more DSPs than the Altera's FFT library. It consumes 6.8% of the available resources for 2^{16}-FFT, and 9.4% for 2^{30}-FFT. Thus, it is feasible. Fig. 11 (c) compares required numbers of ALUTs. Although the implemented FFT consumes more ALUTs than the Altera's FFT library, it consumes only 8.2% for 2^{16}-FFT. Also, it consumes 44.7% for 2^{30}-FFT. Thus, it fits on the modern FPGA. Fig. 11 (d) compares of the maximum clock frequencies. From 2^6-FFT to 2^{14}-FFT, both clock frequencies are comparable. For 2^{16}-FFT, the Altera's FFT library is lower. Since the Altera's FFT library consumes 93.7% of available M9ks, the FPGA has no extra M9k space for the place-and-route. The above results show that, the implemented FFT can realize 2^{14} times wider bandwidth than the Altera's FFT library on the same FPGA. Although the implemented FFT requires additional ALUTs, DSPs, and off-chip DDR2SDRAMs, they are acceptable.

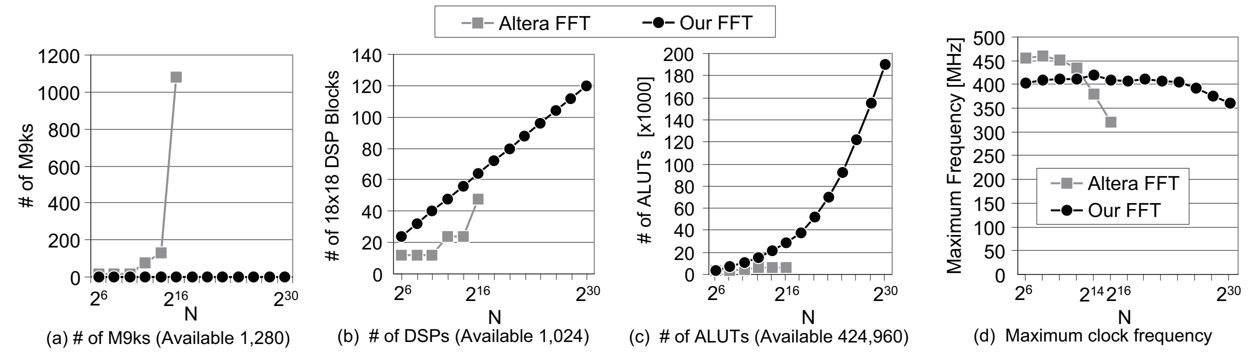

Figure 11: Comparison the proposed FFT with the Altera's FFT.

Table 2: Comparison of GPUs with the proposed method for 2^{27}-FFT.

Method	Time [sec]	Additional Hardware
1GPU	0.442	
2GPU	0.295	Sharing PCI Express
4GPU	0.210	Sharing PCI Express
Proposed Method	0.157	two DDR2SDRAMs

5.5 Comparison with GPUs

With respect to the 2^{27}-FFT for the SETI spectrometer, we compared the implemented FFT with one using the NVIDIA Corp. Tesla S1070 GPU computing system (four GPUs, total VRAM: 16 GB, Power: 800 W) [20]. Table 2 shows that, the implemented FFT is faster than the GPUs. Since the power consumption for the FFT board used for our implementation is less than 10 W, our FFT dissipates much lower power than the GPUs. As for the space, the FPGA-based implementation uses PCI slot, while the Tesla S1070 requires one rack unit. Thus, the FPGA implementation is smaller than the GPU-based one. From above comparison, the FPGA-based one is suitable for the radio telescope. Note that, for the precision, the GPU uses 32-bit floating point, while the FPGA uses 36-bit fixed point. Since the SETI spectrometer adopts 36-bit fixed point as same as our system. From the error analysis, the twice bit length (18 bits in our implementation) of the input (8 bits in our implementation) is enough for the precision in the FFT [5, 17].

6. CONCLUSION

This paper showed the R2^kFFT for a wideband FFT. We analyzed the amount of hardware for the conventional $R2^2FFT$, and showed that the amount of memory is the bottleneck. To reduce the memory, we replaced the twiddle factor memory with the pipelined CORDIC. Also, we extended the radix of the FFT from 2^2 to 2^k in order to reduce the transpose memory. We implemented the 2^{30}-FFT on the Altera's DE4 development and education board. It performs the 2^{30}-FFT in 1.5 seconds. Compared with the Altera's FFT library, our FFT circuit realizes 2^{14} times wider bandwidth using the same FPGA. Also, compared with the Tesla S1070 utilizing four GPUs, our FFT circuit is faster and dissipates lower power. In this way, we implemented the wideband FFT for the radio telescope.

Acknowledgments

This research is supported in part by the grant of Young Researcher Overseas Visits Program for Vitalizing Brain Circulation, and subsidy for Young Researchers by Kagoshima University.

7. REFERENCES

[1] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *FPGA'98*, pp. 191-200, 1998.

[2] Altera Corp., http://www.altera.com/

[3] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex fourier series," *Mathematics of Comput.*, Vol. 19, pp. 297-301, 1965.

[4] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron. Lett.*, Vol. 20, pp. 14-16, 1984.

[5] S. Goedecker, "Fast radix 2,3,4 and 5 kernels for fast fourier transformations on computers with overlapping multiply-add instructions," *SIAM Journal Sci. Compt.*, Vol. 18, pp. 1605-1611, 1997.

[6] HALCA: Highly advanced laboratory for communications and astronomy, http://www.vsop.isas.jaxa.jp/top.html

[7] S. He and M. Torkelson, "A new approach to pipeline FFT processor," *IPPS'96*, pp. 766-770, 1996.

[8] A. Hirota, N. Kuno, N. Sato, H. Nakanishi, T. Tosaki, and K. Sorai, "Variation of molecular gas properties across the spiral arms in IC 342: Large-scale 13CO (1-0) emission", *PASJ*, No. 62, pp. 1261-1275, 2010.

[9] IBOB (Interconnect break-out board), http://casper.berkeley.edu/wiki/IBOB/

[10] D. H. Jones, A. Powell, C. S. Bouganis, P. Y. K. Cheung, "GPU versus FPGA for high productivity computing," *FPL'10*, pp. 119-124, 2010.

[11] H. Karner, et al., "Multiply-add optimized FFT kernels," *Math. Models and Methods in Appl. Sci.*, Vol. 11, pp. 105-117, 2001.

[12] H. Kondo, E. Heien, M. Okita, D. Werthimer, K. Hagihara, "A multi-GPU spectrometer system for real-time wide bandwidth radio signal analysis," *ISPA'10*, pp. 594-604, 2010.

[13] W. R. Knight and R. Kaiser, "A simple fixed-point error bound for the fast fourier transform," *IEEE Trans. Acoustics, Speech and Signal Processing.*, Vol. 27, No. 6, pp. 615-620, 1979.

[14] E. N. Linzer and E. Feig, "Implementation of efficient FFT algorithms on fused multiply-add architectures," *IEEE Trans. Signal Processing*, Vol. 41, pp. 93-107, 1993.

[15] Mars Scout Program, http://www.nasa.gov/

[16] A. Parsons et.al, "PetaOp/Second FPGA signal processing for SETI and radio astronomy," *ACSSC'06*, 2006.

[17] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Prentice-Hall Inc.*, 1975.

[18] SERENDIP: The search for extra terrestrial intelligence at UC Berkeley, http://seti.berkeley.edu/SERENDIP/

[19] SKA: Square kilometer array, http://www.skatelescope.org/

[20] NVIDIA Corp., "TESLA S1070 GPU computing system," http://www.nvidia.com/

[21] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Compt.*, pp. 330-334, 1959.

[22] Xilinx Inc., "LogiCORE IP fast fourier transform v7.1", 2011.

[23] B. Zhou, Y. Peng, and D. Hwang, "Pipeline FFT architectures optimized for FPGAs," *Int'l Journal of Reconfigurable Comput.*, Vol. 2009, 2009.