A New Multichannel Parallel Real-time FFT Algorithm for a Solar Radio Observation System Based on FPGA

Yuan Y. Zhang1, Lei Zhang1, Zi Q. Shang1, Yan R. Su1,2,3, Zhao Wu1,3, and Fa B. Yan1,2,3

1 Laboratory for Electromagnetic Detection, Institute of Space Sciences, Shandong University, Weihai, Shandong 264209, People’s Republic of China
2 School of Mechanical, Electrical & Information Engineering, Shandong University, Weihai, Shandong 264209, People’s Republic of China

Received 2021 November 25; accepted 2022 February 3; published 2022 March 18

Abstract

The real-time fast Fourier transform (FFT) is the essential algorithm for signal processing in a solar radio receiver. However, field-programmable gate array (FPGA) computation resources have become the limitation of real-time processing of signals with increasing time and spectral resolutions. It is necessary to design a real-time parallel FFT algorithm with reduced resource occupation in the development of future receiving systems. In this paper, we developed a multichannel parallel FFT algorithm named the multichannel parallel real-time fast Fourier transform (MPR-FFT), which can greatly reduce FPGA resource occupation while increasing the real-time processing speed. In this algorithm, the $4L$ simultaneous $N$-point FFTs are first converted into $L$ simultaneous $4N$-point FFTs. Fusion processing is then performed to obtain the $4 \times L \times N$-point spectrum. This method has been used in developing a solar radio spectrometer, which works in the frequency range of 0.5–15 GHz in the Chashan Observatory. In this spectrometer, 16 channel MPR-FFT with 8k-point data is realized in a Xilinx UltraScale KU115 FPGA. The MPR-FFT algorithm reduced the computational resources to a large extent compared to the Cooley-Tukey-based parallel FFT method; for instance, the Look-Up-Table, Look-Up-Table RAM, Flip-Flop, and Digital Signal Process slices were reduced by 37%, 50%, 17%, and 2.48%, respectively. Although the MPR-FFT consumes 14 block RAM resources more than the Cooley-Tukey-based parallel FFT, the MPR-FFT algorithm presents an overall reduction in resource usage.

Unified Astronomy Thesaurus concepts: Astronomical techniques (1684); Astronomy software (1855); Radio astronomy (1338); Radio receivers (1355); Radio spectroscopy (1359); Solar instruments (1499); Solar radio telescopes (1523)

1. Introduction

During solar eruptions, i.e., flares, coronal mass ejections (CMEs), etc., magnetic energy stored in the coronal magnetic fields is released and transferred into radiation energy, kinetic energy of energetic particles, and plasma thermal energy. These eruptions may result in severe space weather catastrophes in solar-terrestrial space (Zhao et al. 2004; Hwangbo et al. 2015; Casini et al. 2017; Geng et al. 2018), and the research, monitoring and forecasting of solar eruptions are thus important in the protection of humankind aviation activities.

Radio observations, as one of the two ground-based observation windows, can provide unique information on their source region based on their emission mechanisms, such as plasma emissions at metric wavelengths and gyrosynchrotron emissions at millimetric wavelengths, and are valuable in the study of solar eruptions (Yan et al. 2006; Liu et al. 2011; Casini et al. 2017). Dynamic spectra are usually used as solar radio observations in the decametric to decimetric wavelength regime (Yan et al. 2021). The signal is captured by the observing system by an antenna and transmitted to the analog front-end (AFE) circuit for amplification, filtering, and mixing. The intermediate frequency (IF) signal is then processed by a digital receiver (spectrometer), in which the analog signal is converted into digital signals and yields the real-time dynamic spectrum. The recent development of a fast analog-to-digital converter (ADC) has provided the chance to directly acquire high-frequency signals (several GHz) and could eliminate the mixing process in the AFE circuit (Yan et al. 2020). The application of these high-performance devices could not only improve the time and spectral resolutions of real-time solar radio observations but also reduce the volume, weight and cost of the observation system. However, we should note that fast devices also challenge the design of the digital signal processing module in a receiving system (Luo & Zhang 2020).

The fast Fourier transform (FFT) has been widely used in signal processing in solar radio digital receivers (Morales 2011;
Nakahara et al. 2012; Finger et al. 2013; Nakahara et al. 2016; Iwai et al. 2017; Vaate et al. 2017; Liu et al. 2019). The digital signal converted by ADC is processed by the real-time FFT algorithm to generate the final dynamic spectrum. The serial radix-2 FFT is the commonly used FFT algorithm, in which symmetry and periodicity of discrete Fourier transform (DFT) operations are used. The N-point DFT is first decomposed into N/2 2-point DFTs. The results of these DFTs are then used to yield the FFT value of the initial N-point sequence (Baas 1999).

The serial radix-2 FFT algorithm cannot realize real-time observations with increasing observation data. To solve this problem, many researchers have made improvements to the FFT algorithm, such as the “six-step FFT” algorithm proposed by Bailey (1990) and the Cooley Tukey method (two-dimensional FFT algorithm) (Cooley & Tukey 1965; Deng et al. 2006). In these algorithms, first, perform L simultaneous M-point FFTs on the input data considered an L × M matrix. The resulting data are multiplied by rotation factors and then transposed into an M × L matrix. Finally, M simultaneous L-point FFTs are performed to obtain L × M-point spectrum data (Deng et al. 2006). Zou et al. (2012) optimized the complex multiplier and rotation factors during the implementation of the parallel FFT algorithm. Nakahara et al. (2015) used two techniques to reduce the number of look-up tables (LUTs) in an FPGA. Zhang et al. (2016) developed the Cooley Tukey method and applied it to the broadband and reconfiguration radio observation system for the Five-hundred-meter Aperture Spherical radio Telescope.

However, with increasing observation data, the current parallel FFT algorithm cannot realize real-time observations with limited computational resources. Solar radio bursts could appear in the frequency range of several MHz to tens of GHz. For example, microwave bursts are usually present as continua in the centimetric wavelength regime (Wu et al. 2016, 2019). In this case, observations should be carried out in a frequency range as large as possible, which raises the demands of spectrometers with wider bandwidth together with higher sensitivity. The limited field-programmable gate array (FPGA) computation resource thus restricts the real-time processing of signals in the receiving system, and developing a real-time parallel FFT algorithm with reduced hardware resource occupation is therefore urgent in future radio observations.

Based on the solar radio spectrometer under development (0.5–15 GHz, dual-channel, 14 bit, 3 Giga Samples per second (Gspps)), this paper presents a newly developed multichannel parallel FFT algorithm named the multichannel parallel real-time fast Fourier transform (MPR-FFT). In this algorithm, the 4L channels of data generated from ADC are combined into 2L channels of complex sequences, so 2L IP cores are used to perform simultaneous N-point FFT to generate 2L channels of complex FFT results. The above results are input to the next level to obtain 2L channels of 2N-point real FFTs and then obtain L channels of 4N-point real FFTs. Finally, fusion processing is performed to obtain the 4 × L × N-point spectrum. (L is a positive integer, according to our actual needs, L = 16, N = 512). We tested the MPR-FFT algorithm in a 0.5–15 GHz solar radio digital receiver to realize real-time dynamic spectrum observations with time and spectral resolutions of 1 ms and 366 kHz, respectively. We find that this method can reduce the hardware resource occupation in FPGA to a large extent while increasing the processing speed. The speed improvement refers to the comparison of the time taken by the MPR-FFT algorithm and the Cooley-Tukey-based parallel FFT method to process each frame of data with a working clock of 187.5 MHz. The MPR-FFT algorithm is also suited for navigation, radar, communication, and other fields (Wang & Zhao 2005; Hua et al. 2019). The next section presents the FFT algorithms for parallel processing based on FPGA. Section 3 illustrates the theoretical derivation and implementation in the FPGA for the MPR-FFT algorithm. Section 4 illustrates the experimental results and analysis for the MPR-FFT algorithm, and the discussion and summary are given in the last section.

2. The FFT Algorithms for Parallel Processing

2.1. Parallel Data Processing Method Based on a Single FFT IP Core

FFT IP core is a functional module provided by FPGA manufacturers in the development platform that can be used for FFT calculations. However, due to the strong limitations of FFT IP core, it cannot meet the current needs of astronomical observations. For example, when FFT IP core is used for real-time FFT calculation of pipeline structure, it only supports real-time calculation of single-channel data, which is far from meeting the needs of real-time FFT processing of multichannel parallel data in solar radio telescopes.

Generally, when conducting FFT processing for multichannel data in a single IP core, two operations should be carried out. The parallel-to-serial conversion and clock signal frequency multiplication are shown in Figure 1. As seen from this figure, the signals (S_{i}\text{–}S_{N_{0}}\text{–}1) output from the high-speed ADC are first converted into a serial signal (S_{data}) and then processed by an FFT IP core to yield the spectrum. The data rate of the serial output signal (S_{out}) should be N_{f} times those of input signals, i.e., signal S_{N_{0}}\text{–}S_{N_{0}}\text{–}1, to avoid data loss. Therefore, the processing speed of the IP core should also be N_{f} times the transmitting speeds of the input signals to realize real-time processing. We note that N_{f} cannot exceed 4 in practice, as the IP core has limitations on both processing speed and resource occupation (Deng et al. 2006). This method is not suitable for real-time FFT processing of multichannel parallel data.

2.2. Traditional FFT Algorithm for Parallel Data

At present, parallel FFTs are usually operated based on the “six-step FFT” algorithm proposed by Bailey (1990) and the
Cooley Tukey algorithm (butterfly algorithm). We present in Figure 2 the diagram of the “six-step FFT” algorithm and the Cooley-Tukey-based FFT for a sequence of $N$ points, which can be considered a matrix of $N_1 \times N_2$. FFT IP cores are used to perform $N_1$ simultaneous $N_2$-point pipeline FFTs. Complex multiplications are then carried out for these results and $N_1$ different sets of rotation factors. The full-parallel pipelined FFT is finally processed for the generation of the spectrum.

Thus far, FFT algorithms should balance the processing speed against the occupation of hardware resources in the FPGA. A large amount of resources is occupied by FFTs in each IP core and the full parallel FFT of the $N_1$ points. Realization of real-time processing comes at the cost of hardware resources, such as Look-Up-Table (LUT), Look-Up-Table RAM (LUTRAM), Flip Flop (FF), and Digital Signal Process (DSP) slices. Therefore, it is necessary to design a real-time parallel FFT algorithm with reduced hardware resource occupation in the development of a receiving system.

3. Design of MPR-FFT Algorithm

3.1. Theoretical Derivation of MPR-FFT Algorithm

To realize real-time FFT processing of multichannel parallel data while saving computing resources, we improve and develop a new multichannel parallel real-time FFT algorithm based on the property of odd–even separation in FFT and the principle of the Cooley-Tukey-based parallel FFT algorithm as well as the “six-step FFT” algorithm. In this algorithm, the $4L$ channels of data generated from ADC are combined into $2L$ channels of complex sequences, so $2L$ IP cores are used to perform simultaneous $N$-point FFT to generate $2L$ channels of complex FFT results. The above results are input to the next level to obtain $2L$ channels of $2N$-point real FFTs and then obtain $L$ channels of $4N$-point real FFTs. Finally, fusion processing is performed to obtain the $4L \times N$-point spectrum ($L$ is a positive integer). We will present the derivation of the developed algorithm in the following.

The traditional FFT algorithm (Luo & Zhang 2020) is illustrated as

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

where $k$ is a positive integer in the range of $0 \sim N-1$, $N$ ($N = N_1 \times N_2$) is the length of the FFT, and $n$ can be written as

$$n = n_1N_2 + n_2, \quad (2)$$

where $n_1 = 0, 1, \ldots, N_1 - 1; n_2 = 0, 1, \ldots, N_2 - 1$. That is, the sequence of $x(n)$ is decomposed into $N_1$ sequences, and
each sequence has $N_2$ data, and $k$ can then be expressed as

$$k = k_1N_1 + k_1,$$

(3)

where $k_1 = 0, 1, \ldots, N_1 - 1; k_2 = 0, 1, \ldots, N_2 - 1$. According to the above two equations, we can rewrite Equation (1) as

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} W_{N_1}^{n_1 k_1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] W_{N_2}^{n_2 k_2},$$

(4)

where $\sum_{n_2=0}^{N_2-1} x[n_1, n_2] W_{N_2}^{n_2 k_2}$ is a $N_2$-point FFT calculation and

$$W_{N_1}^{n_1 k_1}$$

is the rotation factor. In this case, the calculation of the $N$-point FFT can be divided into $N_2$-point FFT calculation, multiplication of rotation factors, and $N_1$-point FFT calculation.

We assume a complex sequence as

$$z(n) = c(n) + j \cdot y(n),$$

(5)

where $c(n)$ and $y(n)$ are real sequences. According to the linear property of FFT, the FFT expression for Equation (5) can be written as

$$Z[k] = C[k] + j \cdot Y[k] = Z_r[k] + j \cdot Z_i[k],$$

(6)

where $Z[k]$ is the FFT of $z(n)$ and $C[k]$ and $Y[k]$ are the real part and imaginary part of $Z[k]$, respectively. We can get

$$C[k] = \frac{Z[k] + Z^*[N-k]}{2},$$

$$Y[k] = \frac{Z[k] - Z^*[N-k]}{2 \cdot j},$$

(7)

Equation (7) shows that the FFT results of two $N$-point real sequences can be obtained by the FFT calculation of the $N$-point complex sequence.

The supposed sequence $x(n)$, which is a $2N$-point real sequence, is decomposed into two $N$-point real sequences $h(n)$ and $g(n)$:

$$h(n) = x(2n), n = 0, 1, \ldots, N - 1,$$

(8)

$$g(n) = x(2n + 1), n = 0, 1, \ldots, N - 1.$$  

(9)

A complex sequence $\hat{h}(n)$ can then be expressed as

$$f(n) = h(n) + j \cdot g(n),$$

and its FFT calculation can be written as $F[k] = F_r[k] + j \cdot F_i[k]$. The FFT results (H[k] and G[k]) of $h[k]$ and $g[k]$ can then be derived according to Equation (7). $X[k]$ is the FFT result of the $2N$-point real sequence $(x(n))$, the first half of which is expressed as

$$X[k] = H[k] + G[k] \cdot \exp\left(-j \cdot \frac{\Pi \cdot k}{N}\right) = X_r[k] + j \cdot X_i[k],$$

(10)

$$X_r[k] = \frac{F_r[k]}{2} + \frac{F_r[N-k]}{2},$$

and

$$X_i[k] = \frac{F_i[k]}{2} + \frac{F_i[N-k]}{2}.$$  

(11)

The FFT results of $x(n)$ and $y(n)$ can be obtained (Chen et al. 2009). According to the linear property of FFT, the FFT calculation of sequence $x_{xy}(n)$ is obtained and then illustrated as

$$X_{xy}(k) = X_x(k) + j \cdot X_y(k),$$

(12)

where $X_x(k)$ and $X_y(k)$ are the real and imaginary part of the FFT calculation of $x(n)$, respectively. $X_x(k)$ and $X_y(k)$ are the real and imaginary part of the FFT calculation of $y(n)$. It follows that, $X_{xy}(n) = x(n) + j \cdot y(n), n = 0, 1, \ldots, N - 1,$ and $k = 0, 1, \ldots, N - 1$. According to Formulas 11 and 12, the real part $X_{xy}(k)$ and imaginary part $X_{xy}(k)$ of the first $2N$th elements (FFT calculation of sequence $s(n)$) are obtained and then illustrated as

$$X_{xy}(k) = X_1(k) + \cos\left(\frac{\Pi \cdot k}{2N}\right) X_2(k) - \sin\left(\frac{\Pi \cdot k}{2N}\right) X_2(k),$$

(13)

$$X_{xy}(k) = X_2(k) - \sin\left(\frac{\Pi \cdot k}{2N}\right) X_2(k) - \cos\left(\frac{\Pi \cdot k}{2N}\right) X_2(k),$$

(14)

where $X_1(k) = X_{xy}(k) + X_{xy}(2N-k)$, $X_2(k) = X_{xy}(k) - X_{xy}(2N-k)$, $X_1(k) = X_{xy}(k) + X_{xy}(2N-k)$, $X_2(k) = X_{xy}(k) - X_{xy}(2N-k)$, and assuming $X_{xy}(k) = X_{xy}(k) - X_{xy}(2N-k)$. According to Formulas 16 and 17, the FFT calculation of the
4N-point sequence \( s(n) \) is obtained and then illustrated as

\[
X_s(k) = X_s(k + j * X_d(k)) \quad k = 0 \sim 2N - 1
\]

\[
X_s(k) = \sum_{n=0}^{4N-1} (-1)^n * s(n) \quad k = 2N
\]

\[
X_s(k) = X_s(4N - k)^* \quad k = 2N + 1 \sim 4N - 1
\]

According to Formulas 4, 18, 19 and 20, calculation of the \( N \)-point \((N = N_1 * N_2)\) FFT can be obtained by \( N_1 \) simultaneous \( N_2 \)-point FFT, transforming \( N_2 \)-point complex FFT into \( 4 * N_2 \)-point real FFT, multiplication of rotation factors, and \( N_2 \)-point full-parallel pipelined FFT calculation (fusion calculation). Diagram of algorithm is shown in Figure 3.

Compared with the “six-step FFT” algorithm and the Cooley-Tukey-based parallel FFT, the MPR-FFT algorithm has a remarkable advantage in the processing of large data. The MPR-FFT algorithm can greatly reduce the resource usage in the multichannel simultaneous FFT calculation step and fusion processing step, while the processing speed of the algorithm is improved slightly.

### 3.2. Implementation of the MPR-FFT Algorithm in the FPGA

With the development of ADCs, fast devices have provided the opportunity for multichannel parallel FFT in the field of radio astronomy, communication, radar, etc. FPGA has been widely used in processing parallel digital signals generated by fast ADCs because of its inherent parallelism, flexibility, and integration. In the field of radio astronomy, the FPGA has made it possible to process real-time observations in broad bandwidth. The implementation of multichannel parallel FFT based on FPGA is therefore, worth, researching.

The digital receiver of the 0.5–15 GHz solar radio telescope comprises an ADC sampling module and an FPGA processing module. The board includes two ADC signal acquisition chips, a clock chip, and an FPGA chip. The sampling resolution of ADC is 14 bits, and the sampling rate can be set flexibly from 2.5 to 3.1 Gsps. The clock chip provides high-precision and low-jitter clocks for ADC and FPGA. The FPGA chip is the main controller and performs real-time FFT processing for digital signals. The FPGA device used has dozens of GTH interfaces, which is conducive to signal processing and data transmission. It can provide 1.16 million logical units, 5520 optimized DSP slices, and 16.3 Gbps backplane transceivers.

We present the diagram of signal processing in the receiving system in Figure 4. The analog signal is converted into a digital signal by ADC CORE \( (\text{the sampling rate is 3 Giga samples per second (Gsps)}) \). The digital signal is packaged on the JESD204B interface.

Figure 3. Diagram of the MPR-FFT algorithm for a sequence of \( N \) points that can be processed to a matrix of \( N_1 * N_2 \).

Figure 4. Diagram of signal processing in the receiving system \((L_0-L_7\) are lanes of high-speed links at the JESD204B interface, \(S_0-S_{15}\) are data to be processed every clock cycle, and \(F_0-F_7\) are the output calculations of the FFT algorithm).

Figure 5. Diagram of design architecture of MPR-FFT algorithm for 16 parallel channels with data of 8k-point \((\times0\) to \(\times15\) are 16 channel parallel data from high-speed ADC, \(a_1-a_8\) are calculations of FFT of 512-point complex sequence, \(b_1-b_8\) are calculations of real FFT of 1k-point data, \(c_1-c_4\) are calculations of real FFT of 2k-point data, \(y_1-y_4\) are final generated spectra).
transmission interface on the ADC side and transmitted to the FPGA through the 8 links of the JESD204B LINK (the transmission rate of each link is 7.5 Gbps). The data transmitted by the ADC are received by the JESD204B receiving interface on the FPGA side and unpacked in the FPGA. The data parsed in FPGA are 16 simultaneous parallel channel data sets, so the clock frequency is 187.5 MHz. Here, the calculation method 187.5 MHz × 16 = 3 GHz shows no data loss in this process. Therefore, this fixed clock frequency of 187.5 MHz is the clock frequency of the real-time FFT processing algorithm in the solar radio digital receiver. The 16 parallel sampling data are processed by the real-time FFT algorithm in FPGA with a working clock of 187.5 MHz, and the FFT results are finally transmitted to the host computer through a 10 Gigabit optical fiber.

The MPR-FFT method was carried out in the FPGA for 16 parallel channels with a total data volume of 8k points, and the diagram is shown in Figure 5. The input parallel data (×10) converted by the high-speed ADC are first processed into 8 complex sequences, and complex FFTs of 512-point data are then conducted for these sequences in eight separate FFT IP cores. The above results are used to calculate 2k-point real FFTs (the N–2N module and 2N–4N module in Figure 5), according to which 4 parallel channels obtain 2k points real FFT per channel. Complex multiplications are further carried out for these results and rotation factors. The four-point fusion calculation steps and sorting output modules are finally processed to generate the dynamic spectrum.

3.2.1. Realization of 2N-point real FFT Using the N-point Complex FFT

The 2N-point real FFTs are realized according to N-point complex FFTs by means of calculation-caching calculations. The time sequence diagram is shown in Figure 6. Data_0 and data_1 are the real and imaginary parts of the complex sequence, respectively, which are then processed with a complex FFT of 512-point data. The 2N-point real FFTs are further carried out for these results. We note that the above calculation requires symmetrical operation for the comp_fft of each frame (512 points). Because of the pipelined outputs, the first half-frame signal will be lost when the second half is output and, therefore, should be cached. In the developed method, the cached data are read from the First Input First Output (FIFO) data buffer once the second half frame arrives. As the calculations of symmetric operation for two data points are obtained in one clock cycle, the results of both the first half-frame (out_h) and the second half-frame (out_l) are output in parallel. We should also note that the conversion of results from a complex FFT to a real FFT will result in a time delay of several clock cycles, which would not affect the real-time data processing.

Figure 6. The time sequence diagram of realizing 2N-point real FFT according to the N-point complex FFT in FPGA (data_0 and data_1 are real-time parallel sampling data from ADC, comp_fft is the FFT calculation of 512-point complex sequence, ram_rd is the read signal for extracted data of the first half-frame, the out_h and out_l are results of both the first half-frame and the second half-frame).

Figure 7. Implementation of 4-point fusion processing of the MPR-FFT algorithm (b_1, b_2, b_3 and b_4 are 2k-point real FFTs from the N to 2N module and 2N to 4N module).
3.2.2. Fusion Operation

The results of the $2N$-point real FFT are further processed by a fusion process for the final generation of the spectrum. A diagram of four-point fusion operations is shown in Figure 7. There are four inputs and four reverse outputs in each clock period, and the four outputs of each stage flow directly to the next stage. The input data ($b_1$, $b_2$, $b_3$, $b_4$) are 2k-point real FFTs from the FFT conversion module. Two steps are included in this operation. In the first step, complex multiplication is carried out for the signal of $b_2$ and the rotation factor of the first butterfly operation, the results of which are then added and subtracted with $b_1$ to obtain the signals of $c_1$ and $c_2$. The above operations are also applied to signals $b_3$, $b_4$, $c_3$, and $c_4$. In the second step, complex multiplications are carried out for signals of $c_3$ and $c_4$ and rotation factors of the second butterfly operation, the results of which are then added and subtracted with $c_1$ and $c_2$ to obtain the signals of $d_1$ and $d_2$ and $d_3$ and $d_4$. The register transfer level (RTL) diagram of a butterfly cell is shown in Figure 8, which is one of the four butterfly units in Figure 7.

4. Results

We tested the correctness of the MPR-FFT algorithm and implemented two parallel FFT algorithms (the MPR-FFT algorithm and the Cooley-Tukey-based parallel FFT) in FPGA. Then, the performances of these two algorithms are compared. In our tests, the input data are the sampling data of a single-frequency (2.7 GHz) signal, which are 16 parallel channel data from ADC in the digital receiver with a sampling rate of 3 Gsps. These parallel input data are processed into one channel sequence, and the FFT function is then conducted for this one channel sequence in MATLAB (Figure 9(a)). We also realized a 16 channel MPR-FFT algorithm with 8k points in MATLAB (Figure 9(b)). The results of the regular FFT method (Figure 9(a)) and MPR-FFT algorithm realized in MATLAB (Figure 9(b)) are first compared, and the 16 channel MPR-FFT algorithm is further implemented in FPGA with a working clock of 187.5 MHz (Figure 9(c)). We compared the resource usage and processing speed of the MPR-FFT algorithm and Cooley-Tukey-based parallel FFT. The test results are presented in the following.

Figure 9 presents the FFT results of the regular FFT method (a), MPR-FFT algorithm realized in MATLAB (b), and MPR-FFT algorithm implemented in FPGA (c). We can see that the results of the MPR-FFT method (Figure 9(b)) are identical to those of the regular FFT method (Figure 9(a)). We further implemented the MPR-FFT in FPGA (Figure 9(c)) and found no significant difference among these tests, indicating that the MPR-FFT algorithm can be implemented correctly in FPGA.

Figure 10 presents the simulation results of the MPR-FFT algorithm in FPGA. The pipelined processing ensures real-time processing of multichannel parallel data. The signal power_out displayed as an analog waveform and marked in the white box is the power spectrum of a single-frequency signal of 2.7 GHz, indicating the MPR-FFT algorithm is implemented correctly in FPGA and the real-time solar radio observation is possible.

We carried out tests of Cooley-Tukey-based parallel FFT and the MPR-FFT algorithm in FPGA for 8k-point data, and the corresponding computation resource occupations are shown in Table 1 and Figure 11. We can see that compared with the Cooley-Tukey-based parallel FFT, the MPR-FFT algorithm reduces the LUT, LUTRAM, FF, and Digital Signal Process (DSP) resource occupation of up to 37%, 50%, 17%, and 2.48%, respectively. Although the block RAM (BRAM) resources used in the MPR-FFT are higher than those of the Cooley-Tukey-based parallel FFT, the MPR-FFT algorithm presents an overall reduction in resource usage, and the calculation method will be presented in the following.

In Xilinx UltraScale KU115 FPGA, each slice provides eight 6 input LUTs and sixteen flip-flops. There are two types of slices in the UltraScale architecture. SLICEL (logic) has all the LUT and storage element resources, along with the carry logic and wide multiplexers. A SLICEM (memory) can also use the LUTs as distributed 64 bit RAM. Therefore, the capacity of a LUT is 64 bits. From the eighth row of Table 2, we can see that the amount of LUT for the MPR-FFT algorithm is 8666 less than that of the Cooley-Tukey-based parallel FFT method.
Figure 9. Simulation results of the 8k-point FFT for different processing algorithms. (a) Direct FFT result for 8k points in MATLAB; (b) the simulation result of the 16 channel MPR-FFT algorithm for 8k-point data in MATLAB; (c) the result of the 16 channel MPR-FFT for 8k-point data implemented in FPGA.

Figure 10. The simulation of the MPR-FFT algorithm implemented in FPGA (data_0 to data_15 are the 16 parallel input signals. For the convenience of display, only the signal of data_0 is displayed in analog format. The power_out displayed as an analog waveform and marked in the white box is the power spectrum of a single-frequency signal of 2.7 GHz).

Table 1

| Resource | Utilization of MPR-FFT Algorithm | Utilization of Cooley-Tukey based Parallel FFT | Occupation Rate of MPR-FFT Algorithm | Occupation Rate of Cooley-Tukey based Parallel FFT |
|----------|----------------------------------|-----------------------------------------------|------------------------------------|-----------------------------------------------|
| LUT      | 15422                            | 24088                                         | 2.32%                             | 3.63%                                         |
| LUTRAM   | 4501                             | 9120                                          | 1.53%                             | 3.10%                                         |
| FF       | 48877                            | 58649                                         | 3.68%                             | 4.42%                                         |
| DSP      | 1097                             | 1125                                          | 19.87%                            | 20.38%                                        |
| BRAM     | 94                               | 80                                            | 4.35%                             | 3.70%                                         |
which is equivalent to reducing 541.625 kb of resources. The MPR-FFT consumes 14 BRAM (the capacity of each BRAM is 36 kb) resources more than the Cooley-Tukey-based parallel FFT, which is equivalent to an increase of 504 kb of resources. From the reduced 541.625 kb resources and the increased 504 kb resources, it can be concluded that the MPR-FFT algorithm presents an overall reduction in resource usage compared to the Cooley-Tukey-based parallel FFT method.

In addition to the large reduction in LUT resources consumed, the amount of FF and DSP used in the MPR-FFT algorithm has also been reduced to a certain extent. Depending on the length of the FFT algorithm, the number of LUTs will become a bottleneck; however, that of the BRAMs will not be a bottleneck (Nakahara et al. 2015). Therefore, the significant reduction in LUT resources consumed and an overall reduction in resource usage in the MPR-FFT algorithm are significant improvements.

Table 2 compares resources occupied by each step of the FFT algorithm based on the MPR-FFT and Cooley-Tukey-based parallel FFT methods (the Numbers in Front of the Symbol “/” in the Table are the Resources Occupied in the MPR-FFT Algorithm, the Numbers Behind the Symbol “/” in the Table are the Resources Occupied in the Cooley-Tukey-based Parallel FFT, and the Letters “None” Indicate that there is no Such Step).

| Algorithm Steps                             | LUT  | LUTRAM | FF     | DSP   | BRAM |
|---------------------------------------------|------|--------|--------|-------|------|
| multichannel simultaneous FFT calculation  | 11076/21536 | 4288/8096 | 24688/48545 | 400/800 | 28/56 |
| N to 2N FFT                                 | 2171/None | 0/None | 7080/None | 224/None | 18/None |
| 2N to 4N FFT                                | 330/None | 4/None | 5918/None | 236/None | 4/None |
| multiplication of rotation factors          | 391/6  | 209/0  | 3835/1674 | 96/64  | 12/8  |
| fusion processing                           | 1/1792  | 0/1024  | 6698/8069 | 136/256 | 0/0    |
| Output power spectrum                       | 1453/754 | 0/0    | 658/361  | 5/5    | 32/16  |
| All steps of the algorithm                  | 15422/24088 | 4501/9120 | 48877/58649 | 1097/1125 | 94/80  |

Table 3 compares processing speed between MPR-FFT method and Cooley-Tukey-based parallel FFT of 8k-Point Data (Comparison of the Time Taken by the MPR-FFT Algorithm and the Cooley-Tukey Based Parallel FFT Method to Process Each frame of Data with Working Clock of 187.5 MHz).

| Algorithm Type                  | Number of clocks | Time/μs | Real Time |
|---------------------------------|------------------|---------|-----------|
| Cooley-Tukey based parallel FFT | 1688             | 9002.66 | Real time |
| MPR-FFT algorithm               | 1678             | 8949.33 | Real time |

Figure 11. Proportion of the resources saved in the MPR-FFT compared to the Cooley-Tukey-based parallel FFT algorithm.
Table 3 presents a comparison of the processing speeds of the different FFT methods. The processing speed (starting with the first raw data entering and ending with the last transformation data output) is expressed by the number of clock cycles, and the total time is calculated by a working clock of 187.5 MHz. Table 3 shows that the processing speed of the MPR-FFT algorithm is 10 clock cycles faster than that of the Cooley-Tukey-based parallel FFT. The speed improvement in the paper refers to the comparison of the time taken by the MPR-FFT algorithm and the Cooley-Tukey-based parallel FFT method to process each frame of data with a working clock of 187.5 MHz, and the comparison of the time to process each frame of data is shown in Figure 12. Figure 12 shows that the time for the Cooley-Tukey-based parallel FFT method to process each frame of data (the clock cycles from the beginning of each frame of data to the end of the calculation of one frame of data) is 1688 clock cycles, while the time taken by the MPR-FFT algorithm is 1678 clock cycles. The MPR-FFT algorithm reduces the processing time of each frame of data by 10 clock cycles compared with the Cooley-Tukey-based parallel FFT method, which is equivalent to a 0.5% speed improvement. Because the MPR-FFT algorithm can reduce the complexity of the fusion processing, the processing speed of the MPR-FFT method is improved slightly.

According to the above tests, we can see that there are two significant advancements achieved by the MPR-FFT algorithm:

1. The MPR-FFT algorithm presents an overall reduction in resource usage. The computational resources can be reduced to a large extent, especially the resources of LUT, LUTRAM, and FF, which can effectively solve the resource bottleneck problem.

2. The real-time processing speed of the algorithm is also improved slightly, which guarantees real-time solar radio observations with high time and frequency resolutions in a large bandwidth.

5. Discussion and Summary

The real-time FFT is the essential algorithm for signal processing in a solar radio digital receiver. However, the FPGA computation resource has become the limitation of the real-time processing of signals with increasing time and spectral resolutions in solar radio observation systems. Therefore, it is necessary to design a real-time parallel FFT algorithm with reduced hardware resource occupation in the development of a future receiving system.

We developed a multichannel parallel FFT algorithm named the multichannel parallel real-time fast Fourier transform (MPR-FFT), which could greatly reduce the FPGA resource occupation while ensuring the real-time solar radio observations with high time and frequency resolutions in a large bandwidth. In this algorithm, the 4L channels of data generated from ADC are combined into 2L channels of complex sequences, so 2L IP cores are used to perform simultaneous N-point FFT to generate 2L channels of complex FFT results. The above results are input to the next level to obtain 2L channels of 2N-point real FFTs and then obtain L channels of 4N-point real FFTs. Finally, fusion processing is performed to obtain the 4 * L * N-point spectrum (L is a positive integer). This method would be used in the developing solar radio spectrometer (dual-channel, 14 bit, 3 Giga Samples per second (Gsps)), which would work in the frequency range of 0.5–15 GHz in the Chashan Observatory (CSO).

In addition to the common I–V types of radio bursts, typical radio bursts also include fine-structure bursts such as zebra patterns and spike bursts. In the decimeter-centimeter band observed at 0.5–15 GHz, in addition to the cyclotron synchrotron radiation covering a wider band, highly structured zebra patterns and rapidly changing spike bursts are the main burst structures. Previous studies have shown that the duration of spike bursts is only 10–60 ms (Feng et al. 2018; Tan et al. 2019), and the zebra pattern presents a multilevel harmonic structure in the range of tens to...
hundreds of MHz (Feng et al. 2018; Tan et al. 2019). Therefore, the solar radio telescope needs to have high time resolution and frequency resolution to observe the fine structure of this band. According to the observation requirements of the 0.5–15 GHz solar radio telescope system, the time resolution and frequency resolution of the system are set as 1 ms and 366 kHz, respectively.

In the 0.5–15 GHz solar radio observation system, the signal is divided into multiple analog signal channels, and four digital receivers are used to sample and process these signals in real-time. The sampling rate of two digital receivers is 3 Gsps and that of the other two digital receivers is 2.6 Gsps. When the sampling rate of the digital receiver is 3 Gsps, the instantaneous bandwidth of each acquisition channel is 1.3 GHz. When the sampling rate is 2.6 Gsps, the instantaneous bandwidth of each acquisition channel is 200 MHz. The switching between the multiple analog signal channels is controlled by digital receivers to ensure that solar radio signals of 0.5–15 GHz are all sampled within a period of the time resolution.

In the spectrometer for this observation system, 16 channel MPR-FFT with 8k-point data is realized in a Xilinx UltraScale KU115 FPGA, and significant progress has been achieved in our tests. The computational resources can be reduced to a large extent; for instance, the Look-Up-Table (LUT), Look-Up-Table RAM (LUTRAM), Flip Flop (FF), and Digital Signal Process (DSP) slices are reduced by 37%, 50%, 17%, and 2.48%, respectively. In addition, the real-time processing time to process each frame of data can be reduced by 10 clock cycles working at 187.5 MHz.

Because the 0.5–15 GHz solar radio telescope is still under development, we built a simple platform to verify the performance of the MPR-FFT algorithm running on the digital receiver. In the temporary platform, the existing 2.4 m antenna and turntable in the laboratory are used to receive the solar radio signal of 7.6–8.9 GHz, which is the signal of one analog channel of the 0.5–15 GHz solar radio telescope. In the experiment, an antenna 2.4 m in diameter is used to receive solar radio signals. Amplifiers and filters working in the frequency band of 7.6–8.9 GHz are used to amplify and filter solar signals from the antenna. The processed analog signal is then sent to the digital receiver for digitization and real-time FFT algorithm processing. In the experiment, we recorded the data processed by the digital receiver when the antenna was pointing to the Sun and pointing away from the Sun.

Figure 13 is the spectrum image we processed using the recorded data. The left part of Figure 13 is the result of the antenna pointing to the sky minus its own value, and the right part of Figure 13 is the result of the antenna pointing to the Sun minus the antenna pointing to the sky. Obviously, in the frequency of 7.6–8.9 GHz, the value when the antenna points to the Sun is approximately 1 dB higher than the value when the antenna points to the sky. The digital receiver with the MPR-FFT algorithm can observe the solar radio signal in the experiment, indicating that the MPR-FFT algorithm is suitable for solar radio telescopes.

The 35–40 GHz solar radio telescope that has been built is our very mature observation system. We applied the MPR-FFT algorithm to the digital receiver of the 35–40 GHz solar radio telescope to further verify the performance of the algorithm (Yan et al. 2020; Yan et al. 2021; Shang et al. 2022). The existing 35–40 GHz solar radio telescope has been put into observation for more than 1 yr (Yan et al. 2020; Yan et al. 2021; Shang et al. 2022). At present, its instantaneous bandwidth is 500 MHz, and the output signal of the intermediate frequency bandwidth of the analog receiver is 687.5–1187.5 MHz. In the experiment, the 35–35.5 GHz solar signal, which is the signal of an analog channel of the 35–40 GHz solar observation system, was transformed to 687.5–1187.5 MHz by down-conversion processing. Then, the signal was connected to the digital receiver with the MPR-FFT algorithm for observation.

**Figure 14.** Radio observation experiment in the frequency range of 35–35.5 GHz by using the MPR-FFT algorithm.
As shown in Figure 14, compared with the background values (5° deviate from the Sun), slight and significant enhancements appear when the antenna points to the Sun. We note a phenomenon in the middle of the picture in which the intensity gradually increases every 1.75 s. This results from the rotation of the antenna pointing every 1°, and after rotating the antenna five times, a stable intensity is obtained. It can be concluded that the MPR-FFT algorithm is also suitable for 35–40 GHz solar radio telescopes and has wide applications.

This research was funded by the National Natural Science Foundation of China 41774180, 41904158, the China Postdoctoral Science Foundation (2019M652385), Shandong postdoctoral innovation project (202002004) and Young Scholars Program of Shandong University, Weihai (208220201005).

Professor Yao Chen from the Institute of Space Science of Shandong University provided theoretical guidance and analysis, Jie Liu from the School of Mechanical, Electrical and Information Engineering of Shandong University provided test equipment, and I would like to express my gratitude.

Conflict of interest

The authors declared that they have no conflicts of interest to this work.

ORCID iDs

Yan R. Su @ https://orcid.org/0000-0002-3965-3215
Zhao Wu @ https://orcid.org/0000-0002-6985-9863
Fa B. Yan, @ https://orcid.org/0000-0002-4451-7293

References

Baas, B. M. 1999, IJSSC, 34, 380
Bailey, D. H. 1990, J. Supercomput., 4, 23
Casini, R., White, S. M., & Judge, P. G. 2017, SSRv, 210, 145
Chen, L. J., Yan, Y. H., Liu, F., & Chang, H. 2009, AR&T, 6, 198
Cooley, J. W., & Tukey, J. W. 1965, Macom, 19, 297
Deng, B., Rong, M., & Tang, X. 2006, Computer Engineering, 32, 254
Feng, S. W., Chen, Y., Li, C. Y., et al. 2018, SoPh, 293, 1
Finger, R., Mena, P., Reyes, N., Rodriguez, R., & Bronfman, L. 2013, PASP, 125, 263
Geng, L. H., Tan, C. M., Guo, J. P., et al. 2018, AR&T, 15, 380
Hua, Z., Rong, J., & Luping, X. 2019, PASP, 131, 045002
Hwangbo, J. E., Bong, S. C., Park, S. H., et al. 2015, JASS, 32, 91
Iwai, K., Kubo, Y., Ishibashi, H., et al. 2017, EP&S, 69, 95
Liu, F., Yan, Y. H., Wang, W., et al. 2019, PASA, 36, e043
Luo, Y. Y., Wang, W., Tan, C. M., et al. 2011, AR&T, 8, 248
Ma, Y. J., & Zhang, S. W. 2020, ACES, 48, 922
Morales, M. F. 2011, PASP, 123, 1265
Nakahara, H., Nakamish, H., & Sasao, T. 2012, in Lecture Notes in Computer Science, Algorithms and Architectures for Parallel Processing, ICA3PP (Berlin: Springer), 202
Nakahara, H., Sasao, T., Nakamish, H., et al. 2016, in 2016 IEEE XLVI Int. Symp. on Multiple-Valued Logic (SMVL), ed. L. O’Conner (Sapporo: IEEE), 60
Nakahara, H., Sasao, T., Nakamish, H., & Iwai, K. 2015, in 2015 IEEE Int. Symp. on Multiple-Valued Logic, ed. L. O’Conner (Waterloo, ON: IEEE), 97
Shang, Z. Q., Xu, K., Liu, Y., et al. 2022, ApJS, 258, 25
Tan, B. L., Cheng, J., Tan, C. M., & Kou, H. X. 2019, CB&A, 59, 64
Vante, J., Davidson, D., & Benthem, P. 2017, in 2017 XI European Conference on Antennas and Propagation, ed. L. O’Conner (Paris: IEEE), 3829
Wang, Y., & Zhao, G. 2005, It Age, 25, 5
Wu, Z., et al. 2016, ASTROPHYS. J. LETT, 820, L29
Wu, Z., Chen, Y., Ning, H., Kong, X. L., & Lee, J. 2019, ApJ, 871, 22
Yan, F. B., Liu, Y., Xu, K., et al. 2020, RAA, 20, 226
Yan, F. B., Yang, L., Ke, X., et al. 2021, PASJ, 73, 439
Yan, Y. H., Zhang, J., Chen, Z. J., et al. 2006, AR&T, 3, 91
Zhang, X., Yu, X. Y., Duan, R., et al. 2016, PABei, 34, 249
Zhao, H., Zhan, L., & Rong, Y. 2004, AR&T, 1, 28
Zou, X. B., Liu, Y. X., Zhang, Y. H., et al. 2012, in 2012 VIII Int. Conf. on Wireless Communications, Networking and Mobile Computing, IEEE, 2012, ed. L. O’Conner, 1