IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 2, 2020, accepted February 24, 2020, date of publication March 18, 2020, date of current version March 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981816

Astronomical Data Preprocessing Implementation Based on FPGA and Data Transformation Strategy for the FAST Telescope as a Giant CPS

YUEFENG SONG[1], YONGXIN ZHU[1,2,3], (Senior Member, IEEE), JUNJIE HOU[1], SEN DU[1], AND SHIJIN SONG[1]

[1]School of Microelectronics, Shanghai Jiao Tong University, Shanghai 200240, China
[2]Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China
[3]School of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Yongxin Zhu (zhuyongxin@sari.ac.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1831118, in part by the National Key Research and Development Program of China under Grant 2016YFE0100600, in part by the Shanghai Municipal Science and Technology Commission under Grant 17511105002 and Grant 18511111302, and in part by the National Key Research and Development Program of China under Grant 2017YFA0206104.

ABSTRACT The emergence of cyber-physical-social systems (CPSS) as a novel paradigm has revolutionized the relationship between humans, computers and the physical environment. CPSS extend cyber-physical systems (CPS) to include the social domain, which introduces a challenge of massive data processing. As a typically giant CPS, the Five-hundred-meter Aperture Spherical radio Telescope (FAST), the world's largest filled-aperture radio telescope, generates massive volume of data which poses a huge storage problem that CPSS face likewise and requires real-time data compressing to reduce data storage and movement overhead. The recently introduced Bitshuffle preprocessing algorithm is a novel approach towards exploiting spatial redundancy incorporation to improve the compression ratio with a specific compressor. However, the existing high-performance CPU-based solutions cannot satisfy the performance requirement and power budget requirement simultaneously. In the paper, we propose the implementation of this algorithm on Field Programmable Gate Array (FPGA) and present an unique data transformation strategy to turn raw FAST data in classic FITS format into another format to support huge file sizes, i.e. Hierarchical Data Format (HDF5). Evaluation results show that our implementation can achieve 3.2Gbyte/s throughput which can be equipped with LZ4 compressor to be high performance compressor. This makes Bitshuffle on FPGAs a candidate for meeting the computational and energy efficiency constraints of radio telescopes and provide reference for CPSS facing the same situation.

INDEX TERMS CPSS, astronomical data, FPGA, Bitshuffle, FITS, HDF5.

I. INTRODUCTION

With the cyber-physical system (CPS) technologies evolution [1], lots of interesting application domains have been explored ranging from industry automation to aeronautics and astronautics. Taking human social characteristics into account, an emerging computing paradigm called cyber-physical-social system (CPSS) has focused on the exploration of digital fusion among human [2], computers and CPS since last decade [3]. As an emerging computing paradigm, CPSS is exposed to scale out challenges such as massive data streams arising from diverse data sources.

As a giant CPS data source, radio telescopes face similar challenges [4]. Radio astronomical observation technology is the integration of networks, computing and sharing of data across multiple nodes or systems in terms of the principles of CPS. Unlike traditional CPS devices, the development of the radio astronomy is suffering from challenge of massive

The associate editor coordinating the review of this manuscript and approving it for publication was Zahir Tari.

data [5], [6]. In the case of the Parkes multi-beam sky survey telescope of the 1990s, 2,100 seconds of observations generated data about 100MB in size which is formed into a 1.3GB data file [7]. Under the network transmission conditions at that time, even if 100Mb/s network was used, the data transmission time was shorter than the observation time to obtain data, which could be transmitted from the telescope location in real time. With the development of astronomical observation technology and the improvement of facilities, the scale of astronomical observation data is growing consequently.

The Square Kilometre Array (SKA) radio telescope project is an international project to build an unprecedented scale radio telescope at wavelengths ranging from metre to centimetre. The construction of the Phase-1 SKA telescope (SKA1) will take place from 2022. SKA1 consists of two telescopes namely a low-frequency aperture array (SKA-LOW) and a mid-frequency dish array (SKA-MID). It is estimated that the data rate of SKA1 will approach 1TB/s for the accumulation of SKA-MID and SKA-LOW [8], which requires very high-performance central computing engines and long-haul links with a capacity greater than the global internet traffic. As the precursor facilities of the SKA project, the Australian Square Kilometre Array Pathfinder (ASKAP) consists of 36 identical parabolic antennas, expected to produce 12TB data each day [9], [10]. The Five-hundred-meter Aperture Spherical radio Telescope (FAST) as the world's largest filled-aperture radio telescope currently was declared fully operational on 11 January 2020 [11]. Using a 19-beam receiver with 2GB/s observation data from each beam [12], [13], the FAST must store and maintain a large amount of data that it collects. The traditional data processing methods including data transmission, storage, access and management, have encountered new problems and faced severe challenges when dealing with large scale astronomical observation data [14]. In this case, the data compression algorithm is applied in the field of astronomy. Data compression can play an important role in controlling data scale, reducing network resource consumption, adapted to limited bandwidth and reducing data storage overhead in exchange for avoidance of data overflow [15], thus significantly improve the data processing efficiency when applied for large scale astronomical data. Nowadays, astronomical data compression has been extensively concerned and studied.

The design of an efficient compression algorithm is the basis of the compression scheme. In order to maintain the ordinary observation data in high precision for scientific analysis, the lossless compression algorithms are more valuable. The lossless compression algorithms are functioned based on information entropy and redundancy extraction [16], indicating that the structure and organization form of the astronomical data will have a large impact on the compression performance. The target astronomical data in this research are collected from FAST which contains complex features and multiple dimensions. Therefore, the compression algorithm is supposed to take the data features into consideration, select a customized preprocessing scheme and combine it with classical compression algorithms.

Quite a few researchers improved the predecessors to make the lossless compression algorithms form a developed and complete system. A lossless compression algorithm is achieved by the encoding method to reduce effective storage bits. According to the implementation principle, lossless compression algorithms can be divided into three categories, namely statistical coding, predictive coding and transform coding. Among them, LZ4 is a stable and high-performance compression algorithm making use of statistical coding [17]. The main idea of the algorithm is to find duplicated strings in the input data and replace them with corresponding matches in a reduced bit width. After this, the data is encoded using Huffman tables which can be constructed statically or dynamically. In this research, Bitshuffle preprocessor is equipped with a LZ4 compressor as the foundation to meet the LZ4 performance requirements [18]. A complete compression scheme is evaluated by the compression ratio and compression throughput capacity especially for astronomical big data. The FPGA device can achieve operation acceleration through pipeline technology with an on-chip processing unit. And the FPGA technology is confirmed to be an efficient hardware acceleration approach with the advantage of high acceleration ratio and low power consumption [19]. In our compression preprocessing algorithm, FPGA-based implementation of Bitshuffle fully utilize the acceleration function of FPGA and avoids the loss of throughput when docking with the compressor. In this research, the complete preprocessing procedure is implemented on FPGA and can achieve significant acceleration ratio when compared to the software which is implemented on CPU.

The contributions of this paper can be summarized as:

(1) We proposed an implementation of the Bitshuffle data preprocessing algorithm on FPGAs for data from FAST radio telescope as a giant CPS; (2) We presented a comprehensive analysis of the achieved performance; (3) We present a novel data transformation strategy by using HDF5 for raw FAST observation data to achieve better support for huge file sizes.

The remainder of the paper is organized as follows. The section 2 introduces the background. The section 3 introduces related work. Section 4 describes the Bitshuffle algorithm. Section 5 illustrates the scheme of the transformation of the FAST data format. Section 6 demonstrates the FPGA-based algorithm implementation and optimization. Section 7 shows the experimental results and analyzes them. We draw a conclusion and present the future work in Section 8.

II. BACKGROUND

With the continuous development of information and automation technology, the volume of data has increased fiercely in various industries, and the traditional data storage format for massive data is unable to adapt to the large-scale data storage, especially for radio astronomical data.

FAST currently uses data files in Flexible Image Transport System (FITS) format [20]–[22], which has enjoyed

widespread usage within astronomy for several decades. FITS which is a general approach to encode both a definition of the data and the data itself in a machine independent was adopted on magnetic tape as the standard transport medium. It provides the unambiguous transfer of images that have accuracies up to 32 bits. It supports the transformation for the data type with multi-dimensions. As the advantages of FITS to transmit astronomical images in a standard format were gradually recognized, most major observatories used FITS as the basic format for data exchange. However, data in FITS format can only be accessed by line and its description parts of each observation record is set tightly coupled with the data body where only file read sequentially is supported [22]. Lack of flexibly in layers for datasets management, data failed to be access in parallel and dataset in single file only to be processed in serial make FITS files inadequate for the large amount of astronomical observations collected by FAST.

Several authors have proposed alternative serializations that have advantages over FITS [22]. Motivated by data volumes, HDF5 has been proposed or has been implemented for the LOFAR radio telescope, the CHIME pathfinder [23]. In terms of data size, the HDF5 format can support files in larger than exabyte level. From the aspect of data format definition, more ubiquitous and compatible multidimensional array is adopted in HDF5 with a hierarchical structure similar to the file nested in folder. In terms of file interface, HDF5 provides parallel I/O support after version 2.2.0 [20], [21], which makes data better satisfy the requirement of parallel system. High related heterogeneous data can be put in the same group of HDF5 to facilitate the rapid search of the data. Additionally, HDF5 allows for efficient reading of portions of a dataset, whether they are contiguous or a regular pattern of points or blocks.

The HDF5 file format not only manages datasets in a hierarchical manner, but also divides descriptions of datasets and data body themselves into diverse parts. These advantages make the HDF5 format better than the FITS format for parallel processing of FAST large-scale data. In this paper, we suggest HDF5 as the alternative serialization format for the FITS data model after the discussion about the immediate, practicable advantages of HDF5 and we present the method to transform the data model inherent to the FITS file format to HDF5 in a straightforward manner.

III. RELATED WORK

Many researchers have made extensive and in-depth explorations in the data compression algorithm, especially in the astronomical data or scientific data domain. Masui *et al.* [18] proposed a lossy preprocessing method applied for astronomical radio data. They first adopted a lossy approximation to the data according to the estimated SNR, and then applied a bit-level transpose method to complete the preprocessing, which is confirmed to achieve high compression ratio with acceptable accuracy loss. A widely applicable lossy compression algorithm for scientific multi-dimension datasets was proposed by Tao *et al.* [24] which assumed the data has a polynomial regression characteristic in a sliding window. They performed an efficient predictive encoding method by means of the linear prediction based on the regression relationship. The compression algorithm is able to control the error degree. Amrani *et al.* [25] proposed a lossless compression algorithm for remote-sensing data through wavelet transform methods. The main idea of the research is to extract approximation and detail components in multiple levels and use the approximation components to predict the detail components with a regression model. Another well-known compression algorithm integrated in HDF5 is Szip propose by Yeh *et al.* [26]. The algorithm is based on the extended-Rice algorithm and has high speed and quick adaptation in space statistics. The above algorithms have brought in enlightening ideas and are practical and effective in their specific aspects. However, there still requires a customized algorithm to achieve better performance when applied to the FAST dataset.

In order to enhance the compression speed and data bandwidth, many researches have succeeded to implement compression algorithms on different hardware platforms. Plugariu *et al.* [27] managed to implement a GZIP compression on the FPGA platform. They design the architecture to combine the communication, transmission and operation system for data compression, and achieved high throughput and speed gain. Other algorithms like run-length encoding algorithm were also implemented in hardware, as J. Trein *et al.* described and analyzed [28]. They used on-chip dedicate logic to optimize parallel data processing, and achieved high performance through parallel input. Besides, Banerjee *et al.* [29] proposed the concept of CAM and dictionary-based compression technique and implemented it as an FPGA-based data compressor. The experiment indicated that the design has high processing speed and is suitable for real-time applications.

The experience in the previous researches has reflected the potential and technology feasibility of FPGA based hardware acceleration in astronomical data compression. This research managed to complete the data compression scheme from two aspects of data transformation strategy design and FPGA hardware implementation.

IV. BITSHUFFLE

Bitshuffle, as a preprocessing algorithm available in HDF5, allows optimizing the lossless compression by reordering the data bits in a more compressible order to enhance the lossless coding step performance, which exploits this reduction in entropy to achieve a very high compression ratio [18].

Bitshuffle is subsequently developed after Shuffle algorithm first presented. For typed binary data, where the data consist of arrays of elements of a fixed number of bytes, it has been recognized that compression is generally improved by applying the byte reordering shuffle pre-filter. shuffle breaks apart the bytes of each data element, grouping all the first (second, etc.) bytes. To put this in other terms, if you arrange all the bytes in the array into a matrix with dimensions of

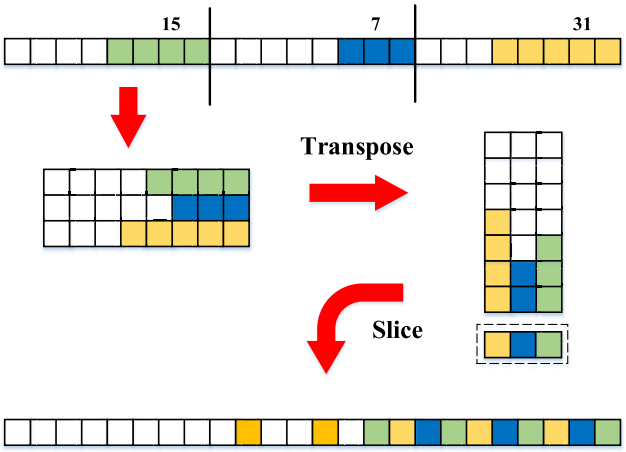


FIGURE 1. Bitshuffle algorithm details and how data bits flow.

the number of elements by the size of each element, shuffle performs a transpose on this matrix.

Bitshuffle extends the concept of shuffle to the bit level: it arranges the bits of a typed data array into a matrix with dimensions of the number of elements by the size of each element (in bits), then performs a transpose. This is illustrated in Figure 1.

If the blank part represents bit 0 and the shaded part represents bit 1, firstly arrange the typical data of the same bit width in the order of physical storage characteristics.

Secondly, similar to byte transposition, corresponding bits of each data element needs to be aligned in order. Then logical view of data bits array normalized is given. Then follows the entire transposed data matrix. The final step is to place the data bits in the output sequence from left to right, then top to bottom, and finally to store them. Bitshuffle is better able to convert spatial correlations into run-lengths than shuffle because it can treat correlations within a subset of the bits in a byte instead of only those which apply to the whole byte.

V. DESIGNING DATA STORAGE FORMATS

As mentioned in Section II, the HDF5 is an open source file format that supports large, complex, heterogeneous data, which uses a file directory like structure that allows to organize data within the file in many different structured approaches.

In order to convert FAST data in FITS format to HDF5 format which is more suitable for large astronomical data processing, this section presents a customized conversion method where the whole information in the original data is entirely retained. The process is completely reversible for all components mapped to the new data format.

A FITS file is comprised of segments called Header-Data Units (HDUs), where the first HDU is called the Primary HDU. Any number of additional HDUs may follow the primary array. These additional HDUs are referred to as FITS extensions.

The FAST telescope is equipped with a 19-beam, dual polarized, cryostat receiver system where each beam can

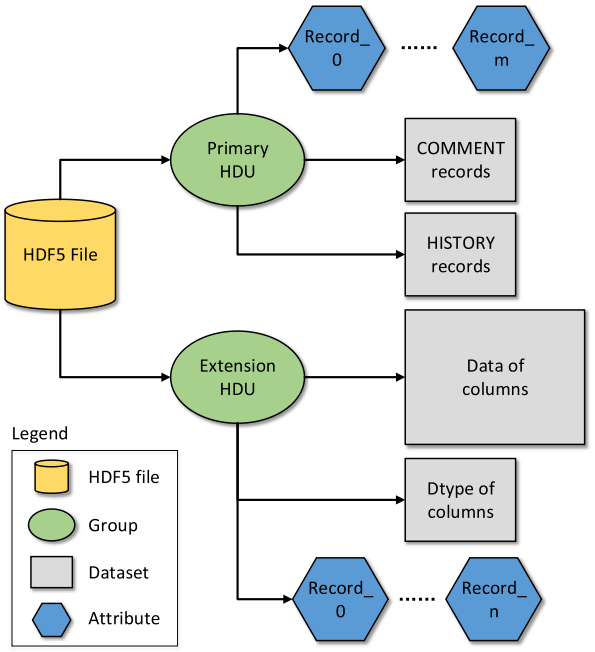


FIGURE 2. A tree diagram describing an example implementation of FAST data transformation from FITS to HDF5.

TABLE 1. Detailed key parameters characterize FAST data.

Key	Value	Comment
BITPIX	8	Number of bits per data pixel
NRCVR	2	Number of receiver polarization channels
OBSNCHAN	4096	Number of frequency channels (original)
NAXIS2	256	Number of rows in table (NSUBINT)
NSBLK	1024	Samples/row (SEARCH mode, else 1)
TFIELDS	17	Number of fields per row

produce 2.1GB of data per second in the drift mode and store it in a FITS file which contains two HDUs. The primary HDU defines the regular file information and context information of the data in the header, without the HDU-DATA section. The second HDU stores the core observations within a column in the DATA part. This is the body of the entire file, as shown in the Figure 2, which is a (256,1024,2,4096) four-dimensional matrix. Each data sampling point represents the power flow density at the specified time-frequency band and polarization.

In order to improve Bitshuffle compressing performance, data are commonly accessed along the time axis. That is, it is generally most efficient for the axis representing time to be the fastest varying once loaded into memory. To meet the requirements for Bitshuffle, data transposition and dimension reduction are needed. The key parameters in FAST data is detailed in Table 1.

CFITSIO is a library of C and Fortran subroutines for reading and writing data files in FITS data format, which

has become the standard widely adopted within the large community of software for astronomers. CFITSIO provides simple high-level routines for reading and writing FITS files that insulate the programmer from the internal complexities of the FITS format. CFITSIO also provides many advanced features for manipulating and filtering the information in FITS files. CFITSIO was released as an ANSI-C version after FITSIO was originally written in Fortran-77. Callable CFITSIO wrappers were then packed as libraries in various programming languages like C, C sharp, Python and R, *et al.*, to provide interfaces for CFITSIO access. We select the python3.7 version as programming approach.

The progress is as follows: Firstly, create the hdf5 file to get the file handle that will serve as the top-level directory for the entire hierarchical storage mode. The file consists of two groups used to preserve the contents of the primary HDU and the second HDU, respectively. Every record is fetched except COMMENT and HISTORY records, with key concatenated by a specific prefix to identify the original name of the current field. At the same time, in order to satisfy possible order requirements, the sequence number is spliced into the wrapped name. The approach needs to be applied to the comment portion of each record likewise. For COMMENT and HISTORY records in the header, since there may be multiple items in the two records, this means that we need to scan the entire HDU-header section to ensure the integrity of COMMENT and HISTORY. Specifically, taking HISTORY as an example, we need to save one or more histories into an array, then encode them with ASCII, and save them in the form of the dataset to the group corresponding to primary HDU. The advantage of this method is that the information in the COMMENT and HISTORY can be searched and updated instantly, and the data can be effectively tracked to update the change record through the burying point, thus ensuring data backtracking and tracking.

The conversion of the header part has been completed above. The core data of FAST observation is collected in the DATA part of the second HDU, which contains 17 columns. All the entries within a given column of an ASCII or Binary table extension have the same datatype. However, datatypes vary from column to column and are required to be specified when memory space is requested, which means datatypes in all columns need to be retained in transformation to ensure data recovery. So, we record the type information for the 17 columns of data through the DATA_TYPE field and store it in the second group. Next step, by default, we place 17 columns of data in the group as 17 datasets, and the keywords and sequence numbers must be recorded as attributes in the corresponding datasets. If data of specific frequency band or polarization need to be accessed instantly, the column containing real observation data can be divided into distinct datasets in HDF5 output file.

In terms of how HDU is arranged in the HDF5 file. As HDF5 allows hierarchy, then the groups of related HDUs are allowed to be stored in the same layer. It is required that each HDU is attached to the root group to keep the compatibility between two file formats.

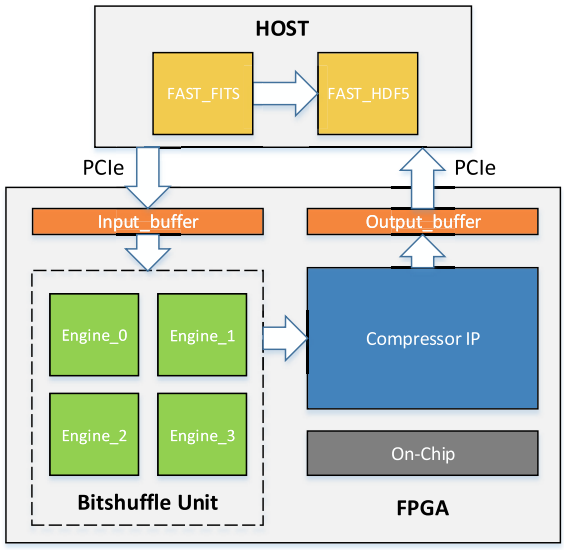


FIGURE 3. System architecture showing data format transformation, host-kernel communication, Bitshuffle preprocessing and compressing.

To be compatible with legendary program libraries, the storage format for FAST was standardized by astronomy scientists with the metadata description and the data body in the classic FITS format instead of HDF5. Additional standardization work has been done in this paper for modern format HDF5. HDU in FITS is placed within a corresponding HDF5 group, with each record in HDU written as an attribute into a group to document the metadata. The core data body is split into multiple datasets attached by a file handler or one group, depending on the data access method. Nonetheless, the values of the data contained within the original file are required to be unchanged.

VI. FPGA-BASED HARDWARE DESIGN

In this section, we discuss implementation details of the Bitshuffle algorithm on FPGA and show how accelerator engines are integrated into hardware architecture to realize a complete system theoretically.

To evaluate the feasibility of the hardware system accelerating Bitshuffle algorithm, the FPGA implementation of the algorithm has been carried out in the form of multiple engines design with the optimized modules, which fully utilizes the data throughput and operation capability of FPGA.

FPGA can potentially deliver tremendous acceleration in high-performance servers and embedded computation applications with large scale arithmetic logic units and pipeline design technology. In this work, in consideration of better performance and flexibility of the low-level design approach of the language, we introduce languages on RTL level to describe the algorithm on our design.

In our design, we first evaluate the parallelization feasibility of the operations units, and then continue to distribute

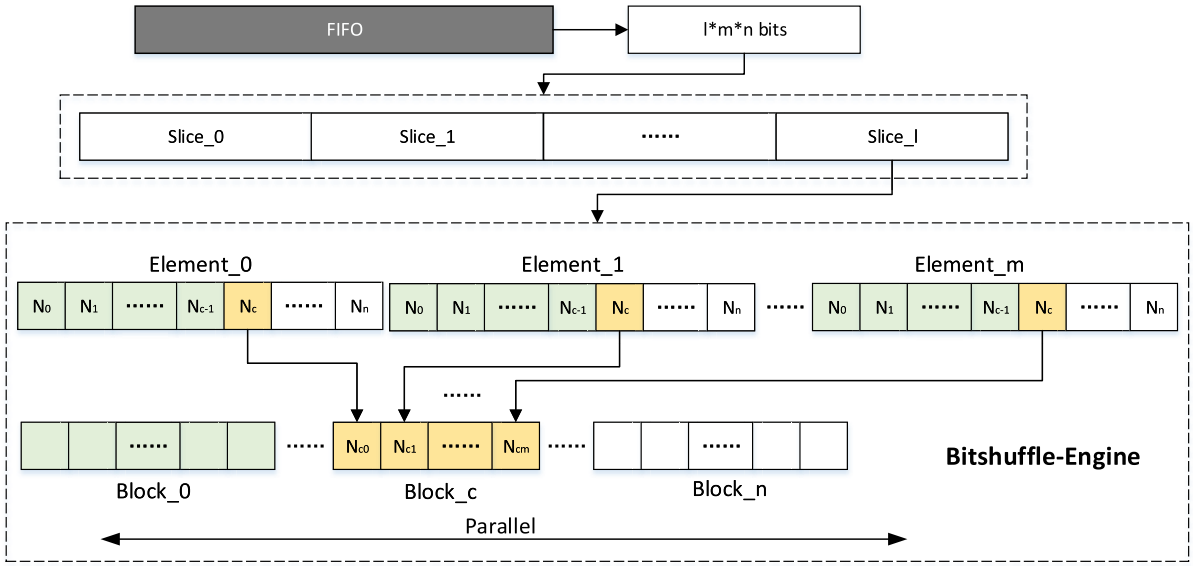


FIGURE 4. Framework of Bitshuffle engine showing how data elements flow and how data outputted from FIFO sliced.

them to the host and FPGA devices. The complete system architecture design is shown in Figure 3.

The total system architecture consists of two parts: the CPU-based host and the FPGA-based device. The host is responsible for transferring data in the form of FITS to HDF5 and delivering data between the host and the FPGA device via PCIe. The mainstream PCIe-based CPU-FPGA platforms use direct memory access (DMA) for an FPGA to access the data from a CPU. The FPGA typically needs a memory controller IP to read the data from the CPU's DRAM to its own DRAM through PCIe [30]. In fact, this communication is limited by restrict bandwidth in practice to make it impractical to implement full-speed acceleration even if we have a high throughput preprocessing accelerator on the FPGA side. In this design, high-bandwidth PCIe device is required to eliminate the potential throughput bottleneck to ensure data flow transmission in maximum volume.

FAST data of specified frequency channel and polarization is transmitted to FPGA with time dimension as the sequential reference, and an instantiated FIFO is customized to utilize block RAM to receive this part of data which supports up to 1024 bits. The maximum data transfer efficiency supported by FIFO should not be a bottleneck in the total system. Data in buffer needs to be handed over to the Bitshuffle engine for processing, and larger data bandwidth requires multiple Bitshuffle engines.

To improve the compression throughput and overall system efficiency of the FPGA Bitshuffle accelerator design, we exploit a multi-way parallel design, where each accelerator engine can process a relatively small amount of data concurrently. The framework of Bitshuffle engine is shown in Figure 4. Data then is handed over to the LZ4 compressor for further compression. The last step is that the output data will be buffered in the output buffer while waiting to be transferred via DMA.

Due to pieces of data located in nearby regions on the timeline in a file typically share similar ranges of values. The power flux density data at specific frequency channel and polarization is designed to be successively divided into pieces of the size of (l*m*n) bits in a time sequence and then delivered into FIFO. l represents the number of Bitshuffle engines, which is equal to the number of slices taken from FIFO in one cycle; m represents the number of the typed data element in the engine; n refers to the bit width of each element. Note that fetched (l*m*n) bits data is hard to process, in consideration of limited hardware calculation capacity in one clock and the potential requirement for higher computing frequency, throughput should be restricted precisely.

After taken over by engine, multiple slices fetched during one clock will be processed concurrently. According to the output element position, the computation unit will locate the corresponding bit of target in the original data and sets it. Similarly, this process is executed in parallel and outputs n m-bits blocks.

VII. EXPERIMENT RESULTS AND ANALYSES

After the theoretical analysis of the Bitshuffle and FPGA implementation of the preprocessing algorithm, this section concentrates on validating data transformation and evaluating the complete compression scheme through simulations and experiments on different platforms. The experiment results confirm that the work not only shows the feasibility of the algorithm in specific astronomical dataset, but also achieves significant improvement through FPGA implementation.

A. DATA TRANSFORMATION

Radio astronomy data is required to be transformed from FITS format into HDF5, which is used as the raw data format before data preprocessing. The size of original FITS file for one beam observation by FAST is 2,172,683,520 bytes. Besides, the size of HDF5 file created by transformation is 2,172,683,520 bytes with additional information of COMMENT and HISTORY cards updated which has just increased by less than 0.002%. However, the integrity of the data is guaranteed and that data access interface is equipped with more superior features, such as efficient reading of portions of datasets, parallel I/O support, etc.

B. EXPERIMENTAL DATASET

The data set of this work is the radio astronomy data collected and quantified by the Five-hundred-meter Aperture Spherical radio Telescope (FAST), which is derived from the PSR B1257+12 pulsar. The pulsar is a millisecond pulsar, which is a research hotspot in astronomy and has important significance in research on exploring the evolution process of celestial bodies, analyzing the composition and state of matter, etc. It has been observed successfully by astronomical telescopes due to the pulsed particle radiation, which has become one of the important discoveries in the field of astronomy.

The target observation data was collected by the central beam among 19 beams of the FAST radio telescope in drift mode, which indicates that the signal has typical characteristics and is classified as the representative strong pulsar signal. It contains the pulsar signal distribution characteristics with a certain universality. After the sampling and quantification process of the national observatory, the original data set contains four dimensions, which respectively represent the following information:

- The first two dimensions represent the time sample block and the time sample point in each sample block, respectively.
- The third dimension represents the polarization information of the radio signal, including two polarization terms.
- The fourth dimension represents the flux density signifying signal strength collected by each frequency channel.

Since the limited buffer depth of input data and the large dataset, we select the Block RAM as an approach for data initialization on the chip in the form of a COE file, in order to evaluate the total hardware acceleration process started from the buffer queue staging data.

In order to keep the sequential order of the data, 4 dimensions of data set is reduced to 3 dimensions which contains time domain, frequency distribution and the two polarizations. And the data are stored in the type of 8-bit unsigned integer (uint8). Bitshuffle is applied according to the internal correlation and redundancy. The data value distribution of the original dataset is shown in Figure 5.

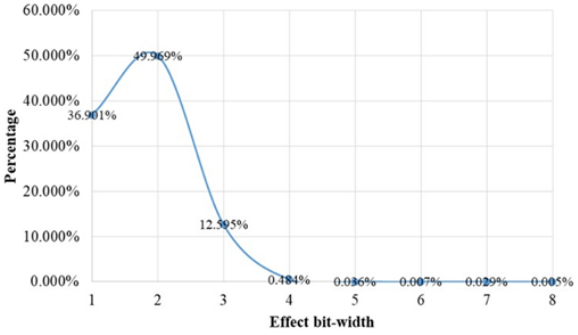


FIGURE 5. Data value distribution in grey-scale map.

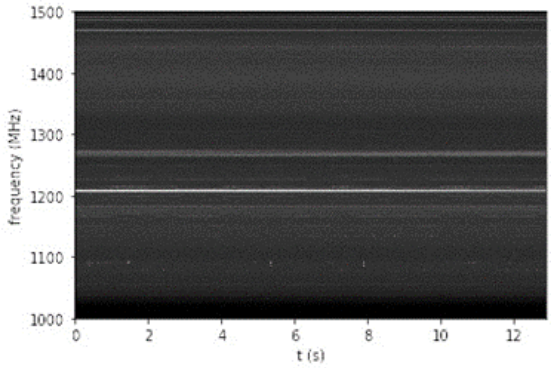


FIGURE 6. The effect bit-width distribution of the dataset.

From the Figure 5, it is evident that the data value reaches a peak around a specific frequency band, and presents an attenuation trend in the time domain. Since the data are stored in the bit-width of 8 bits, we made a further step to collect the effect bit width (defined as the minimum bit-width to cover the value range) of the dataset. The result is shown in Figure 6.

According to the two figures, the original astronomical data presents the feature that most data are component with low effect bit-width. Additionally, the data volume on the time dimension is 10 times that on the frequency dimension, since data accessed along the time axis typically helps improve Bitshuffle compressing performance.

C. EXPERIMENTAL SETUP

Although current work is aimed at the evaluation on the hardware acceleration effect of Bitshuffle preprocessing at this stage, considering that the implementation of LZ4 on FPGA still needs to consume a lot of resources, we chose the accelerator with abundant resources on chip as far as possible. The target platform we select is Xilinx Virtex UltraScale+ VCU1525 FPGA where I/O resource and block RAM are sufficient and 16-lane PCIe connector is implemented which performs data transfer at the rate of 8.0 GigaTransfers per second at maximum [31], since Bitshuffle on FPGA is an I/O-intensive application and FPGA-based LZ4 is a compute-intensive application.

TABLE 2. Optimal parameters of maximum performance for Bitshuffle processing.

Parameters	Value
Number of engines (l)	8
Number of elements each engine (m)	2
Bit width of each element(w)	4096
Feasible FIFO maximum bit width (b)	256
Clock frequency(f)	1024

D. BITSHUFFLE ACCELERATOR PROCESSING

The generalized hardware implementation method is given in the previous section, without specific parameters. However, reasonable parameters can make full use of hardware resources to improve the speed performance, so we need as fully as possible the use of buffer queue data bandwidth. The maximum bandwidth of the FIFO is 1kbit/s, supporting the throughput of 12.8GB/s in 100MHz working frequency. Unfortunately, as single Bitshuffle engine in both input and output needs to be a large amount of I/O resources, once the number of tasks an engine processing and the number of engines created exceeds the ceiling board of FPGA resources, failure of the experiment will be led. Through repeated experiments, we give a reasonable parameter configuration scheme, as follows in Table 2.

E. RESULTS AND ANALYSIS

Our scheme is for the Bitshuffle algorithm targeting data preprocessing before compressing, and the implementation effect is needed of the current optimal LZ4 compressor as a reference to infer whether the performance of the compressor integrated with Bitshuffle accelerator can reach the theoretical optimal result. We refer to the optimal implementation effect of LZ4 based on FPGA. Xilinx FPGA-based LZ data-compression architecture contains multiple compression engines which run concurrently to get higher throughput. Each compression engine is designed to process 1byte/clock with a clock frequency of 250MHz. So if design contains N(8) compression engines, overall throughput will be Nx250MB/s ($8 \times 250 = 2$GB/s) [32]. For Bitshuffle parts, each compression is designed to process 16byte/clock with a frequency of 100MHz. Dual engines created will achieve 3.2GB/s for overall throughput.

We also measure the throughput performance of a CPU-based implementation with basic X86 instructions and another implementation scheme of Bitshuffle compiled with SSE2 (Streaming SIMD Extensions 2) respectively [18] in comparison with FPGA-based implementation. CPU we selected is Intel Core i7-3770 at a 3.40GHz clock rate. Comparison results shown in Table 3 indicate that Bitshuffle implemented on FPGA outperforms the state-of-the-art implementations based on CPU.

Table 4 shows that I/O accounts for the majority of the power consumption, which indicates that FPGA-based Bitshuffle is an I/O intensive application. Table 5 illustrates the resource utilization for our Bitshuffle design on the VCU1525 FPGA, which demonstrates that multiple engines executing in parallel is limited by the over-utilization problem.

TABLE 3. Performance (Throughput) of FPGA implementation compared with CPU implementation with basic X86 instructions and CPU implementation compiled with SSE2 instructions.

Platform	Instruction Set	Frequency	Throughput
Intel Core i7-3770	X86-64	3.40GHz	17MByte/s
	SSE2	3.40GHz	767Mbyte/s
Xilinx VCU1525	NULL	100MHz	3.2GByte/s

TABLE 4. Report of power.

On-Chip	Power (W)	Used	Utilization (%)
LUT as Logic	0.406	128	0.01
Register	0.076	256	0.01
BUFG	<0.001	1	0.42
I/O	99.708	515	76.18

TABLE 5. Report of utilization.

Site Type	Used	Util%
Bonded IOB	515	76.18
HPIOB_M	240	76.92
HPIOB_S	240	76.92
HPIOB_SNGL	35	67.31
CLB LUTs	128	0.01
CLB Registers	256	0.01

VIII. CONCLUSION

To handle huge data stream processing challenges, the pending issue in the emerging CPSS paradigm, we proposed the FPGA-based prototype of Bitshuffle preprocessing algorithm with a classic LZ4 compressor. We also presented an approach for data file format transformation from FITS to HDF5, which is able to support huge files. This work would be the leading attempt to accelerate Bitshuffle on FPGA for radio telescope observation data processing. Through the analysis of the algorithm behavior and bottlenecks, we customized and optimized the system architecture and computing logic to maximize the bandwidth efficiency and computing performance. The Bitshuffle prototype achieves 3.2GB/s performance on the target FPGA.

This work demonstrates the performance optimization strategy of Bitshuffle on the target FPGA board. The optimizations presented in the paper can be further applied to future higher performance hardware design, such as Xilinx Alveo and achieve even higher performance.

Additionally, from the research on the preprocessing and compression of massive data flow oriented to FAST, unprecedented methods to deal with the processing challenges of

mass data stream can be generalized and transferred among other application scenarios of CPSS.

ACKNOWLEDGMENT

The authors would like to thank the Xilinx University Program (XUP) for their support of Xilinx EDA tools and hardware. They are very grateful for the warm reception and skillful help that they have received. Our deepest gratitude goes to the editor and anonymous reviewers for their careful work.

REFERENCES

[1] J. Zeng, L. T. Yang, M. Lin, H. Ning, and J. Ma, "A survey: Cyber-physical-social systems and their system-level design methodology," *Future Gener. Comput. Syst.*, vol. 105, pp. 1028–1042, Apr. 2020.

[2] G. Xiong, F. Zhu, X. Liu, X. Dong, W. Huang, S. Chen, and K. Zhao, "Cyber-physical-social system in intelligent transportation," *IEEE/CAA J. Automatica Sinica*, vol. 2, no. 3, pp. 320–333, Jul. 2015.

[3] E. K. Wang, Y. Ye, X. Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow, "Security issues and challenges for cyber physical system," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput. (CPSCom)*, Dec. 2011, pp. 733–738.

[4] T. M. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, and M. A. Díaz-Bouza, "A fog computing based cyber-physical system for the automation of pipe-related tasks in the Industry 4.0 shipyard," *Sensors*, vol. 18, no. 6, p. 1961, Jun. 2018.

[5] D. L. Jones, K. Wagstaff, D. R. Thompson, L. D'Addario, R. Navarro, C. Mattmann, W. Majid, J. Lazio, R. Preston, and U. Rebbapragada, "Big data challenges for large radio arrays," in *Proc. IEEE Aerosp. Conf.*, Mar. 2012, pp. 1–6.

[6] X. Cai, L. Pratley, and J. D. McEwen, "Online radio interferometric imaging: Assimilating and discarding visibilities on arrival," *Monthly Notices Roy. Astronomical Soc.*, vol. 485, no. 4, pp. 4559–4572, Jun. 2019.

[7] D. G. Barnes, F. H. Briggs, and M. R. Calabretta, "Postcorrelation ripple removal and radio frequency interference rejection for Parkes Telescope survey data," *Radio Sci.*, vol. 40, no. 5, pp. 1–10, Oct. 2005.

[8] H. Barwick, "SKA telescope to generate more data than entire Internet in 2020," Computerworld Australia, North Sydney, NSW, Australia, Tech. Rep., 2011. [Online]. Available: https://www.computerworld.com/article/3484960/ska-telescope-to-generate-more-data-than-entire-internet-in-2020.html

[9] G. Hampson *et al.*, "ASKAP PAF ADE—Advancing an L-band PAF design towards SKA," in *Proc. Int. Conf. Electromagn. Adv. Appl.*, 2012, pp. 807–809.

[10] A. R. Duffy, M. J. Meyer, L. Staveley-Smith, M. Bernyk, D. J. Croton, B. S. Koribalski, D. Gerstmann, and S. Westerlund, "Predictions for ASKAP neutral hydrogen surveys," *Monthly Notices Roy. Astronomical Soc.*, vol. 426, no. 4, pp. 3385–3402, 2012.

[11] R. Nan, "Five hundred meter aperture spherical radio telescope (FAST)," *Sci. China Ser. G*, vol. 49, no. 2, pp. 129–148, Apr. 2006.

[12] D. Li, P. Wang, L. Qian, M. Krco, A. Dunning, P. Jiang, Y. Yue, C. Jin, Y. Zhu, Z. Pan, and R. Nan, "FAST in space: Considerations for a multibeam, multipurpose survey using china's 500-m aperture spherical radio telescope (FAST)," *IEEE Microw. Mag.*, vol. 19, no. 3, pp. 112–119, May 2018.

[13] H. Li, J. Sun, G. Pan, and Q. Yang "Preliminary running and performance test of the huge cable robot of FAST telescope," in *Cable-Driven Parallel Robots* (Mechanisms and Machine Science), vol. 53, C. Gosselin, P. Cardou, T. Bruckmann, and A. Pott, Eds. Cham, Switzerland: Springer, 2018.

[14] W. B. March, K. Czechowski, M. Dukhan, T. Benson, D. Lee, A. J. Connolly, R. Vuduc, E. Chow, and A. G. Gray, "Optimizing the computation of n-point correlations on large-scale astronomical data," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, pp. 1–12.

[15] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[16] A. F. Heavens, R. Jimenez, and O. Lahav, "Massive lossless data compression and multiple parameter estimation from galaxy spectra," *Monthly Notices Roy. Astronomical Soc.*, vol. 317, no. 4, pp. 965–972, 2002.

[17] A. Farruggia, P. Ferragina, and R. Venturini, "Bicriteria data compression: Efficient and usable," in *Proc. Eur. Symp. Algorithms*, 2014, pp. 406–417.

[18] K. Masui, M. Amiri, L. Connor, M. Deng, M. Fandino, C. Höfer, M. Halpern, D. Hanna, A. D. Hincks, G. Hinshaw, J. M. Parra, L. B. Newburgh, J. R. Shaw, and K. Vanderlinde, "A compression scheme for radio data in high performance computing," *Astron. Comput.*, vol. 12, pp. 181–190, Sep. 2015.

[19] M. Bartik, S. Ubik, and P. Kubalik, "LZ4 compression algorithm on FPGA," in *Proc. IEEE Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2015, pp. 179–182.

[20] D. C. Price, B. R. Barsdell, and L. J. Greenhill, "HDFITS: Porting the FITS data model to HDF5," *Astron. Comput.*, vol. 12, pp. 212–220, Sep. 2015.

[21] K. Anderson, A. Alexov, L. Bähren, J.-M. Grießmeier, and M. Wise, "LOFAR and HDF5: Toward a new radio data standard," in *Proc. Astronomical Data Anal. Softw. Syst. Astronomical Data Anal. Softw. Syst.*, 2011, p. 53.

[22] E. W. Greisen, "FITS: A remarkable achievement in information exchange," in *Information Handling in Astronomy—Historical Vistas*. Dordrecht, The Netherlands: Springer, 2003.

[23] E. W. Greisen, M. R. Calabretta, F. G. Valdes, and S. L. Allen, "Representations of spectral coordinates in FITS," *Astron. Astrophys.*, vol. 446, no. 2, pp. 747–771, Feb. 2006.

[24] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May/Jun. 2017, pp. 1129–1139.

[25] N. Amrani, J. Serra-Sagristà, V. Laparra, M. W. Marcellin, and J. Malo, "Regression wavelet analysis for lossless coding of remote-sensing data," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 9, pp. 5616–5627, Sep. 2016.

[26] P.-S. Yeh, W. Xia-Serafino, L. H. Miles, B. Kobler, D. A. Menascé, and J. H. Day, "Implementation of CCSDS lossless data compression in HDF," in *Proc. Earth Sci. Technol. Conf.*, 2002.

[27] O. Plugariu, A. D. Gegiu, and L. Petrica, "FPGA systolic array GZIP compressor," in *Proc. 9th Int. Conf. Electron., Comput. Artif. Intell. (ECAI)*, Jun. 2017, pp. 1–6.

[28] J. Trein, A. T. Schwarzbacher, B. Hoppe, and K.-H. Noffz, "A hardware implementation of a run length encoding compression algorithm with parallel inputs," in *Proc. IET Irish Signals Syst. Conf. (ISSC)*, 2008, pp. 337–342.

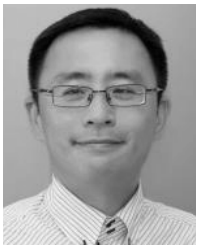
[29] T. P. Banerjee, A. Konar, and A. Abraham, "CAM based high-speed compressed data communication system development using FPGA," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 959–964.

[30] W. Qiao, J. Du, Z. Fang, M. Lo, M. C. F. Chang, and J. Cong, "High-throughput lossless compression on tightly coupled CPU-FPGA platforms," in *Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2018, pp. 37–44.

[31] H. Kavianipour, S. Muschter, and C. Bohm, "High performance FPGA-based DMA interface for PCIe," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 2, pp. 745–749, Apr. 2014.

[32] *Xilinx LZ4*. Accessed: Nov. 8, 2019. [Online]. Available: https://github.com/Xilinx/Applications

YUEFENG SONG received the B.S. degree in mechanical and electronic engineering from the Changchun University of Science and Technology, in 2013, and the M.S. degree in mechanical and electronic engineering from the University of Electronic Science and Technology of China, in 2017. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include high-performance computing, big data, and machine learning.



YONGXIN ZHU (Senior Member, IEEE) joined the Shanghai Advanced Research Institute, Chinese Academy of Sciences (CAS), as a Full Professor, in 2017. He is also an Adjunct Professor with the School of Microelectronics, Shanghai Jiao Tong University (SJTU). Prior to his tenure with CAS and SJTU, he worked as a Research Fellow with the National University of Singapore, from 2002 to 2005, a Senior Consultant with S1 Incorporation (inventor of the 1st Internet banking in the world), from 1999 to 2002, and a Teaching Assistant with the Department of Computer Science and Engineering, SJTU, from 1994 to1995. He was also a Visiting Professor with the National University of Singapore, from 2013 to 2017. He has published over 150 English journal articles and conference papers, 50 Chinese journal articles, and 20 Chinese patent approvals in the areas of computer architecture, embedded systems, big data processing, and blockchain. He has received over 20 million RMB in grants from various funding agencies and industrial partners in China. He is a Distinguished Member of China Computer Federation (CCF) and Blockchain Technical Committee of CCF. With over 1000 citations of these works in recent years, he has received recognition in China and Asia with the IEEE Best Paper Awards and Shanghai Innovation Award. He is a Guest Editor of *Journal of Systems Architecture*.

JUNJIE HOU received the B.S. degree in mechanical and electronic engineering from the Changchun University of Science and Technology, in 2013, and the M.S. degree in mechanical and electronic engineering from the University of Electronic Science and Technology of China, in 2016. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include high-performance computing and big data.

SEN DU received the bachelor's degree in electronic science and technology from Wuhan University, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include edge computing and customized computing.

SHIJIN SONG received the bachelor's degree from the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, in 2017. She is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. Her current research interests include object detection, machine learning, and neutral networks.