Mechanical and Aerospace Engineering
B. Guan (Ed.)

doi:10.3233/ATDE250059

An Onboard Heterogeneous Computing Framework Compatible with the Satellite Management System

Qing LAN [a,b,c] , Lu LI [a,b,c,1], Zhijiang WEN [a,b,c], Junwang HE [a,b] and Wen CHEN [a,b]
[a] *Innovation Academy for Microsatellites of Chinese Academy of Sciences, Shanghai, 201306, China*
[b] *Shanghai Engineering Center for Microsatellites, Shanghai 201306, China*
[c] *University of Chinese Academy of Sciences, Beijing 100039, China*
ORCiD ID: Qing Lan https://orcid.org/0009-0000-3693-9202
ORCiD ID: Lu Li https://orcid.org/0009-0002-8665-0612
ORCiD ID: Zhijiang Wen https://orcid.org/0000-0001-9102-9324

Abstract. In recent years, satellites have exhibited a trend toward increased intelligence, leading to the need for the on-orbit deployment of artificial intelligence (AI) algorithms. Intelligent applications require heterogeneous computing to achieve accelerated processing. At the same time, employing a centralized onboard computing system can reduce the overhead associated with intra-satellite data interaction and management. In this context, this paper proposes an onboard heterogeneous computing framework compatible with the satellite management system. The design employs a chip-level heterogeneous architecture, utilizing CPU, GPU, and FPGA as the computational units. To manage and utilize the heterogeneous computing resources, the satellite management software structure, based on a multi-core processor, is expanded to form a heterogeneous computing software framework. This framework includes dependency libraries for intelligent applications and introduces an intelligent application management mechanism. The satellite management system, along with a database and intelligent application, is deployed on a hardware platform based on the Yulong810A multi-core heterogeneous processor. Test results demonstrate that the satellite management software, database, and intelligent applications function cooperatively and complete computational tasks within the required timeframes, thereby validating the fundamental capabilities of the proposed onboard heterogeneous computing system.

Keywords. Onboard heterogeneous computing, satellite management system, intelligent applications, multi-core processor

1. Introduction

As the number of satellites continues to increase and their functionalities become more complex, the demand for intelligent satellite systems is also growing rapidly. Numerous researches have emerged focusing on algorithms for autonomous satellite operation and management, such as autonomous mission scheduling[1][2][3] and onboard resource management[4][5][6].

[1] Corresponding Author: Lu Li, lilu16@mail.ustc.edu.cn

However, artificial intelligence (AI) algorithms are primarily computationally intensive tasks that often involve extensive matrix operations[7][8], making them well-suited for Single Instruction Multiple Data (SIMD) processing. Onboard CPUs typically utilize a Single Instruction Single Data (SISD) architecture, and even when upgraded to multi-core CPUs, they generally follow a Multiple Instruction Multiple Data (MIMD) architecture. CPUs are more suitable for handling logic-intensive tasks, but their peak computational power is significantly lower than that of GPUs, which employ a SIMD structure, under the same transistor scale. Therefore, a CPU-only onboard computing system is insufficient to meet the high-performance computing demands of intelligent applications. Heterogeneous computing, which combines multiple types of processors, is required to achieve computational acceleration and ensure both performance and flexibility in the computing system.

Heterogeneous computing is a specialized form of parallel computing that utilizes different types of computational units within a single system. Heterogeneous computing can be implemented at various levels, including system-level, board-level, and chip-level heterogeneity. System-level heterogeneity involves selecting and configuring different computational units and interconnecting them via external interfaces, while board-level and chip-level heterogeneity are achieved through on-chip interconnections of different types of processors. Heterogeneous computing integrates general-purpose computing resources like CPUs, multipurpose computing resources like FPGAs, specialized computing resources like ASICs, and storage resources into a unified resource pool, providing the computational power needed for various types of complex high-performance computing tasks.

Given that current mature onboard processors are mostly single-core[9] and have relatively low computational power, existing onboard computing systems typically adopt a distributed design to increase computing resources by adding more computers. For example, the satellite management computer may be paired with a payload management computer, or an AI processing computer[10]. However, multiple computing devices result in higher communication and management overhead. With the enhancement of onboard computing unit performance[11][12], a single computer can now provide sufficient computing power, making it possible to manage both the satellite management system and the data processing system on a single computer through centralized management. Nonetheless, the satellite management system requires high real-time performance and low computational load, whereas data processing demands lower real-time performance but higher computational capacity. Therefore, the hardware architecture must be designed to minimize computational latency and provide scalability, while the software architecture must ensure that the satellite management software and data processing software work collaboratively without interfering with each other.

In this paper, we propose an onboard heterogeneous computing system compatible with the satellite management system. The design employs a chip-level heterogeneous architecture, utilizing CPU, GPU, and FPGA as the computational units. To manage and utilize the heterogeneous computing resources, the software structure of the satellite management system, based on a multi-core processor, is expanded to form a heterogeneous computing software architecture.

2. Onboard Heterogeneous Computing Hardware Architecture

2.1. Typical Computational Units

Typical computational units include CPUs, FPGAs, GPUs, and ASICs. The CPU is characterized by its extensive control logic, branch prediction, and other functions, while its arithmetic logic unit (ALU) occupies only a small portion of the chip area. This makes CPUs the most flexible and easiest to program, making them suitable for general-purpose computing and control tasks.

FPGAs, on the other hand, are distinguished by their ability to dynamically alter their hardware logic structure through software, allowing for task-specific optimization to achieve higher computational efficiency. FPGAs offer abundant computational resources and strong parallel processing capabilities, along with stable and extremely low latency. This makes FPGAs well-suited for streaming processing and applications requiring low latency, as well as for computation-intensive and communication-intensive tasks.

GPUs are notable for their large number of cores, often numbering in the thousands, which support high concurrency and offer powerful floating-point computation capabilities. While GPUs are less flexible than CPUs, they generally outperform FPGAs in executing general algorithms, making them ideal for numerical computations involving large volumes of homogeneous data and computation-intensive tasks.

ASICs feature circuits that are customized according to specific tasks and algorithms, meaning the chip logic cannot be modified. Although ASICs have a long development cycle, they outperform other computational units in power efficiency and performance in specialized fields, making them suitable for dedicated computational tasks.

A comparative summary of these typical computational units is provided in Table 1.

Table 1. Comparison of typical computing units.

Architecture	Performance	Latency	Power Consumption	Flexibility
CPU	Low	Very High	High	Very High
GPU	High	High	Very High	High
FPGA	High	Extremely Low	Low	High
ASIC	Very High	Extremely Low	Low	None

2.2. Hardware Architecture Design

The organizational structure of a heterogeneous computing system typically centers around a CPU, with other types of computational units used as accelerators, forming a CPU+xPU heterogeneous architecture. The selection of the appropriate level of heterogeneity and types of computational units is determined by the task requirements and the characteristics of the computational units, forming the basis of the heterogeneous computing system. For an onboard heterogeneous computing system, the primary consideration is real-time performance, where computational latency must be minimized to meet the needs of real-time satellite control and operations. The second consideration is generality, as onboard processors must undergo extensive validation for in-orbit operation, meaning that the chips should not be limited to a few specific tasks. Future satellites will need to employ various types of intelligent algorithms to accomplish functions such as mission scheduling, fault prediction, and image analysis, requiring onboard computing resources to be general-purpose, flexible, and scalable.

To reduce computational latency, a chip-level heterogeneous approach is adopted, integrating the computational units on a single chip, allowing all units to directly access memory. This eliminates the need for memory copying and storing data in local memory during inter-unit data exchange, saving significant time. Given that FPGA's DMA functionality is well-developed, additional FPGA boards can be added via PCIe interfaces to expand computing resources when necessary. Considering the needs of various intelligent algorithms and the requirement for flexibility, a multi-core CPU, GPU, and FPGA Fabric are selected as the computational units on the chip. The CPU is responsible for task management and handling general-purpose computations with low computational demands, the GPU is used for tasks such as image processing or matrix operations, and the FPGA Fabric is utilized for task-specific computational acceleration. The hardware architecture of the heterogeneous computing system is illustrated in Figure 1.

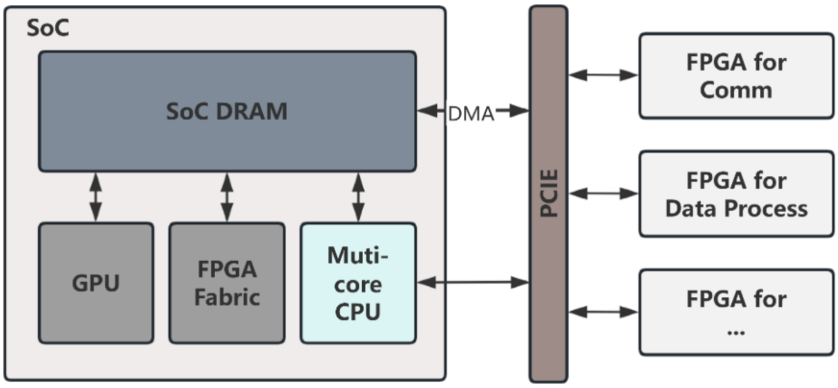


Figure 1. Hardware framework of heterogeneous computing system.

To achieve chip-level heterogeneity by integrating CPU, GPU, and FPGA Fabric on a single chip, chiplet technology is selected[13]. Chiplets are pre-manufactured, function-specific die components that can be combined and integrated. Within a single package, these chiplets are interconnected using on-chip interconnect technology, forming a complete chip with complex functionalities. The use of chiplet technology will allow for the rapid integration of various types of computational units, reducing both the cost and time required for chip design.

3. Onboard Heterogeneous Computing Software Architecture

3.1. Overview

To manage and utilize heterogeneous computing resources, the software architecture based on a multi-core processor needs to be expanded, forming a heterogeneous computing software architecture. In order to be compatible with the satellite management system, the satellite management software and advanced applications that depend on heterogeneous computing must run simultaneously on the same computing platform. First, the operating system needs to be able to provide all the management functions required for both to run. Meanwhile, since the satellite management software is a real-time periodic application, while advanced applications such as AI algorithms are typically non-real-time, their operation should not interfere with each other.

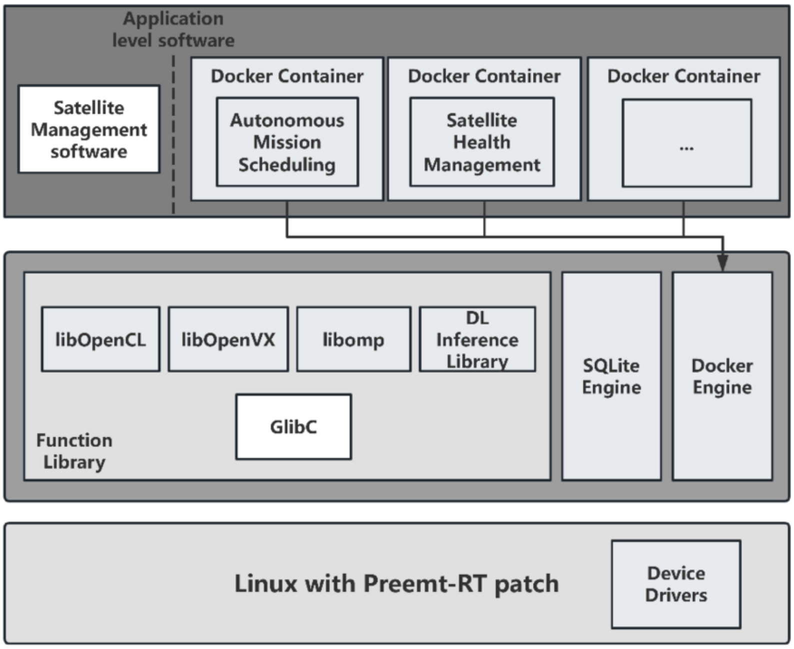


Figure 2. Structure of heterogeneous computing software.

To solve the above problems, a software structure as shown in Figure 2 was designed. On the platform software side, the primary components include loading drivers for the heterogeneous computing units within the operating system, adding libraries that intelligent applications depend on, designing a heterogeneous computing scheduler, and introducing an intelligent application management mechanism. Additionally, a database is used for unified management and integration of onboard data.

On the application software side, the primary focus is on extending the functionalities of the satellite management software that is compatible with multi-core processors to include advanced applications such as database and intelligent algorithms, organizing them in an appropriate manner.

3.2. Low Level Software

In terms of the operating system, Linux is used as the operating system due to the extensive system functionality required to support heterogeneous computing and intelligent applications. The Linux configuration and customization should consider real-time performance, expandability and support for intelligent applications, so Linux with Preempt-RT patch[14] is chosen with only the necessary drivers loaded.

Regarding libraries, executing programs on different types of computing devices requires the use of the OpenCL heterogeneous programming framework. Device programs are written in a C-like language and depend on the OpenCL core library, “libOpenCL”, for execution. While the satellite management software based on multi-core processors uses task distribution across different CPU cores for parallel computing, intelligent applications often require breaking down a task into multiple cores for simultaneous computation to increase speed, necessitating parallel models like OpenMP, which relies on the “libomp” library.

For software development related to image processing and analysis, the cross-platform computer vision library OpenCV provides a useful programming interface. If hardware acceleration is required for computer vision processing, OpenVX can be used to optimize performance and power consumption, with dependencies on the “libopencv” and “libOpenVX” libraries, respectively. For deep learning applications, an efficient and user-friendly deep learning inference framework is necessary, one that supports various types of neural network architectures and mainstream neural network model storage formats. NCNN, an open-source neural network inference library developed by Tencent and optimized for mobile devices, is a suitable choice[15]. This library has no third-party dependencies and is fully implemented in C++, making it easy to port across different platforms. It also has a small size of less than 700KB, which is advantageous for onboard deployment. Therefore, NCNN can be used as the onboard deep learning inference framework, although other frameworks can be selected based on specific application needs.

Regarding task scheduling, tasks on the CPU can be distributed to different CPU cores through CPU affinity settings, with satellite management software and intelligent applications assigned to different CPU sets for execution, isolating them to prevent the operation of intelligent applications from impacting the satellite management software. Since intelligent applications are primarily non-real-time, while satellite management software is real-time, isolating real-time and non-real-time applications helps ensure the real-time performance of the satellite management software. For tasks on other computing devices, tasks should be allocated to the appropriate computing device based on the task type and the characteristics of the computing device, with tasks on these devices typically following a first-in, first-out (FIFO) scheduling method.

3.3. Application Level Software

The development and usage of intelligent applications differ in several ways from that of satellite management software. Firstly, intelligent applications are often developed by multiple organizations, each responsible for their own applications, leading to varying requirements for the runtime environment, such as different deep learning inference frameworks. Secondly, unlike satellite management software, which is rarely updated in orbit, intelligent applications require more frequent updates and injections due to the rapid pace of upgrades and the discrepancies between in-orbit application scenarios and ground simulation environments. To enhance compatibility and management capabilities, Docker container technology[16] is introduced to manage intelligent applications. Docker provides a virtual runtime environment with far lower system resource overhead than virtual machines, allowing intelligent applications to run in the same environment on the satellite as during ground development, without requiring additional configuration. This saves time and effort in synchronizing production environments. Moreover, Docker isolates and limits system resources, running intelligent applications in a sandbox, so that the startup, shutdown, or failure of an application does not affect other applications in the system, facilitating in-orbit updates of intelligent applications. To use Docker, the

Docker engine needs to run within the platform software, providing support for Docker containers, while intelligent applications run within these containers.

Intelligent applications often require the fusion of various types of information as input. For instance, mission scheduling applications need to gather information on satellite power, storage space, current attitude, and tasks to be executed, while fault prediction applications require operational status data from various satellite devices. On the satellite management software side, as software functionality increases, the data interactions between modules become more complex, and there is a need to manage and back up operational status and mode data for various satellite subsystems. Therefore, a database management system is needed to manage and integrate the large and complex data onboard. SQLite can be used as the database management system[17]. SQLite is a lightweight, open-source relational database widely used in embedded devices on the ground and can be reduced to a size of less than 250KB. Additionally, the SQLite engine is not an independent process but is linked to the application itself, resulting in low operational latency and overhead, making it well-suited for use in space.

In terms of application software, since Linux supports multi-process operations, various applications on the satellite can run and be managed independently as separate processes. Besides satellite management software, the application software on the onboard heterogeneous computing system primarily consists of various intelligent applications, including autonomous mission scheduling, satellite health management, and onboard resource management. The core of these intelligent applications lies in the neural network structures and models. To deploy a trained neural network on the satellite, the network must be quantized and pruned to reduce its size and computational load, and it must be deployed in a C/C++ environment, retaining only the data processing and forward propagation steps.

The heterogeneous computing software architecture designed in this paper can effectively manage and utilize heterogeneous computing resources while supporting the operation of both satellite management software and intelligent applications, providing a foundation for satellite intelligence.

4. Application and Testing

4.1. Testing Scheme

To validate the basic functionality of the heterogeneous computing system described in this paper, the satellite management software and extended intelligent applications were deployed on a multi-core heterogeneous hardware platform for operational testing.

In terms of hardware, the multi-core heterogeneous development board Yulong810A-DKIT[18], developed by Orbita company, was selected as it largely meets the requirements of the heterogeneous computing hardware architecture described in this paper. This development board employs chip-level heterogeneity, featuring a quad-core CPU and an AI coprocessor composed of a GPU and a neural network acceleration unit. It supports interfaces like OpenCL/OpenVX and can seamlessly integrate with mainstream deep learning software frameworks such as TensorFlow and Caffe.

In terms of software, the Linux with Preempt-RT patch is used, with the addition of satellite management software, a database management system and an intelligent application example program provided by Orbita. The example program uses the YOLOv3-tiny algorithm for ship target detection and utilizes Orbita's YLANN as the

deep learning inference framework. Although Docker facilitates the deployment of intelligent applications, it is not essential for basic functionality verification; hence, Docker is not used in this validation process. The basic functionality of the heterogeneous computing system is verified through the successful operation of the satellite management software and intelligent applications, as well as data management via the database.
The testing process is as follows:

- Database Porting: Since SQLite has no third-party dependencies, it is compiled using a cross-compiler to generate the SQLite library "libsqlite3".
- Modification of Intelligent Application: The intelligent application example program is modified to allocate its usable CPU set to Core 3, isolating it from the satellite management software. Additionally, SQLite database access operations are added, allowing the application to write its computational results to the database.
- Modification of satellite management software: The satellite management software is modified by adding SQLite database access operations in the task management module. Each cycle, the software reads the computational results of the intelligent application from the database and prints them on the console.

The collaborative testing scheme for the heterogeneous system application is illustrated in Figure 3.

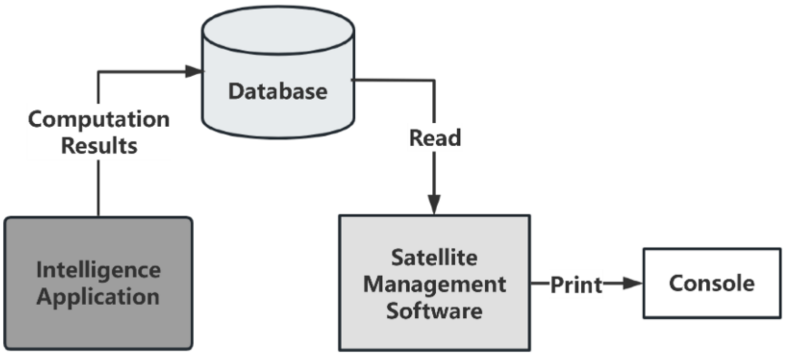


Figure 3. Collaboration scheme of heterogeneous system application test.

4.2. Test Results

The operational test was conducted by running the system continuously for 1000 seconds. During this time, the average execution time per cycle of the satellite management software was recorded, along with the average time taken by the intelligent application to complete target detection in a single 512x512 image. Additionally, the console output was monitored.

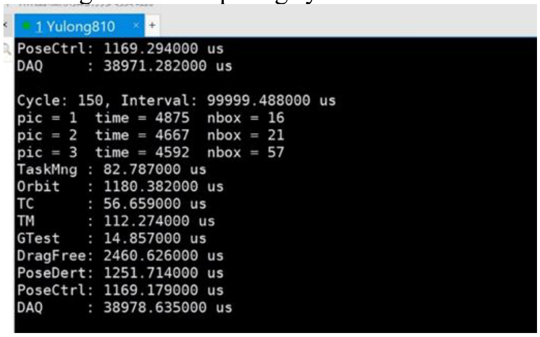
The average execution time of the application software is shown in Table 2. The results indicate that, even under high computational demands, the satellite management software was able to complete each cycle within the required deadline, with sufficient idle time remaining. This suggests that the system has the capacity to handle more complex tasks in the future.

The intelligent application took an average of 4573 microseconds to complete target detection in a 512x512 image. This demonstrates that the intelligent application can perform its computations within a reasonable timeframe, indicating that the heterogeneous computing system possesses strong computational capabilities, sufficient to support certain onboard intelligent applications.

Table 2. Execution time of the application software.

Application Software	Attribute	Average Execution Times(us)	Average response jitter(us)	Dependency
Satellite Management Software	A Cycle	50321	1.89	libm,librt,libdl, libsqlite
	B Cycle	42684	2.01	
	C Cycle	42704	1.03	
	D Cycle	42749	1.15	
	E Cycle	50273	0.70	
	F Cycle	52688	0.76	
	G Cycle	50293	1.21	
	H Cycle	52729	1.19	
	I Cycle	42691	1.36	
	J Cycle	42693	1.43	
Intelligent Application	YOLOv3-tiny Algorithm	4573	/	YLANN

The console output is shown in Figure 4, which indicates that the satellite management software successfully read the data written by the intelligent application from the database. This confirms that the satellite management software, database, and intelligent application are functioning together as intended, thereby validating the basic functionality of the heterogeneous computing system described in this paper.



```
PoseCtrl: 1169.294000 us
DAQ     : 38971.282000 us

Cycle: 150, Interval: 99999.488000 us
pic = 1  time = 4875  nbox = 16
pic = 2  time = 4667  nbox = 21
pic = 3  time = 4592  nbox = 57
TaskMng : 82.787000 us
Orbit   : 1180.382000 us
TC      : 56.659000 us
TM      : 112.274000 us
GTest   : 14.857000 us
DragFree: 2460.626000 us
PoseDert: 1251.714000 us
PoseCtrl: 1169.179000 us
DAQ     : 38978.635000 us
```

Figure 4. Console output

5. Conclusion

This paper extends the satellite management system towards an onboard multi-core heterogeneous computing system. The design employs a chip-level heterogeneous architecture, utilizing CPU, GPU, and FPGA as the computational units in the hardware framework. To manage and utilize the heterogeneous computing resources while ensuring that the satellite management software and data processing software work together without interference, an onboard heterogeneous computing software architecture is designed. The system's functionality was validated through operational experiments by deploying the satellite management software, expanding the database, and running intelligent applications on a multi-core heterogeneous hardware platform. The test results demonstrate that the satellite management software, database, and intelligent applications function cooperatively and complete computational tasks within the required timeframes, thereby confirming the basic functionality of the onboard heterogeneous computing system described in this paper.

References

[1] Dalin L, Haijiao W, Zhen Y, Yanfeng G and Shi S 2020 IEEE Geoscience and Remote Sensing Letters 18 1901–1905

[2] Huang Y, Mu Z, Wu S, Cui B and Duan Y 2021 Remote Sensing 13 2377

[3] Lam J T, Rivest F and Berger J 2019 International Conference on Theory and Practice of Natural Computing (Springer)pp 184–196

[4] Lin Z, Ni Z, Kuang L, Jiang C and Huang Z 2024 IEEE Transactions on Wireless Communications

[5] Hu X, Wang L, Wang Y, Xu S, Liu Z and Wang W 2022 IEEE Communications Letters 26 808–812

[6] Lin Z, Ni Z, Kuang L, Jiang C and Huang Z 2022 IEEE Transactions on Vehicular Technology 71 3917–3930

[7] Girshick R 2015 Proceedings of the IEEE international conference on computer vision pp 1440–1448

[8] Han K, Wang Y, Chen H, Chen X, Guo J, Liu Z, Tang Y, Xiao A, Xu C, Xu Y et al. 2022 IEEE transactions on pattern analysis and machine intelligence 45 87–110

[9] He J, Xu L, Xu D, Yu S, Wang K and Chang L 2020 2020 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD) (IEEE) pp 352–356

[10] Zhang X, Chen W, Zhu X, Meng N, He J, Bi X, Zhang Y, Shi Q, Li F, Liu R et al. 2024 Science China Technological Sciences 67 240–258

[11] Hijorth M, Aberg M, Wessman N J, Andersson J, Chevallier R, Forsyth R, Weigand R and Fossati L 2015 DASIA 2015 DAta Systems in Aerospace 732 7

[12] Zhang C, He X, Zhan P, Qi Z, Gu M and Yan D 2020 Communications, Signal Processing, and Systems: Proceedings of the 8th International Conference on Communications, Signal Processing, and Systems 8th (Springer) pp 2068–2074

[13] Li T, Hou J, Yan J, Liu R, Yang H and Sun Z 2020 Electronics 9 670

[14] Reghenzani F, Massari G and Fornaciari W 2019 ACM Computing Surveys (CSUR) 52 1–36

[15] Peng Y and Wang Y 2021 Computers and Electronics in Agriculture 187 106253

[16] Merkel D et al. 2014 Linux j 239 2

[17] Bhosale S, Patil T and Patil P 2015 Int. J. Comput. Sci. Mob. Comput 44 882–885

[18] Tang H, Ye B, Shi H, Wang H and Yu T 2023 3rd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2023) vol 12717 (SPIE) pp 282–289