Energy-Efficient Computing Acceleration of Unmanned Aerial Vehicles Based on a CPU/FPGA/NPU Heterogeneous System

Xing Liu, Wenxing Xu, Qing Wang, and Mengya Zhang

***Abstract*—The time and energy optimization of computationally intensive tasks involving unmanned air vehicles (UAVs) is highly important for increasing the reaction speed of UAVs and for prolonging their lifetime. To achieve the above objective, many studies based on heterogeneous computing have been carried out. Although these studies have achieved good results, limitations remain. First, neural processing units (NPUs) have emerged in recent years. However, insufficient attention has been devoted to CPU/NPU research in academia currently. Second, most popular heterogeneous computing architectures have only one kind of accelerator, e.g., CPU/GPU or CPU/field programmable gate array (FPGA). A heterogeneous system with multiple kinds of accelerators, e.g., CPU/FPGA/NPU, has not been investigated in depth. To address the above concerns, we propose a heterogeneous CPU/FPGA/NPU system aimed at realizing energy-efficient computing acceleration for computationally intensive UAV tasks. First, we select several representative computationally intensive UAV tasks and design FPGA and NPU accelerators dedicated to these tasks. Then, we calculate the time and energy costs of these tasks on the FPGA and NPU, respectively, and find that different tasks are appropriate for running on different cores. Based on this finding, we further build a heterogeneous CPU/FPGA/NPU architecture and assign each UAV task to the most appropriate core for execution. In this way, the UAV tasks can be executed more efficiently. We conduct experiments by executing all the representative UAV tasks on the CPU, CPU/GPU, CPU/FPGA, CPU/NPU and CPU/FPGA/NPU platforms. The results show that a heterogeneous system with multiple accelerators can achieve better computing performance and higher energy efficiency.**

***Index Terms*—Completion time, energy, heterogeneous computing, unmanned aerial vehicles.**

I. Introduction

CURRENTLY, unmanned air vehicles (UAVs) have been widely applied in many potential military and civilian applications, such as environmental surveillance, disaster relief, agricultural services, and battlefield reconnaissance [1]. On UAVs, many computationally intensive tasks, such as deep neural networks, fast fourier transform (FFT), and image processing, need to be executed to enhance their capabilities. Since these tasks have high-computational complexity, they prolong a UAVs decision-making time and also decrease its lifetime.

To address the above challenge, it is important to optimize the completion time and energy cost for computationally intensive UAV tasks. In recent years, many research efforts have been made to achieve this objective, and heterogeneous computing is a significant approach that has been widely applied to solve this problem [2]. Heterogeneous computing refers to a system that is equipped with more than one kind of computing core. Typically, a heterogeneous computing system consists of one general-purpose CPU, which acts as the host, and one or more types of accelerators, such as graphics processing units (GPUs), field programmable gate arrays (FPGAs), and neural processing units (NPUs). Since different computing cores in heterogeneous systems have different hardware architectures, they are, therefore, appropriate for performing different kinds of tasks [3]. At runtime, each UAV task can be assigned to the core which is the most suitable for executing it. In this way, both the computing performance and energy efficiency of UAVs can be improved.

In recent years, many studies have been devoted to using CPU/GPU and CPU/FPGA heterogeneous computing architectures to accelerate computationally intensive tasks on resource-constrained embedded systems, and they have achieved good results. However, these works still have some limitations. First, NPUs have emerged as a new computing architecture in recent years, and many NPU processors have been developed, such as Huawei's Ascend NPU [4], [5], Cambrian's DianNao series [6], [7], [8], [9], and Google's TPU [10]. These NPUs demonstrate excellent performance in executing artificial intelligence (AI) algorithms, yet CPU/NPU heterogeneous computing has not received sufficient research attention in academia compared with the CPU/GPU and CPU/FPGA heterogeneous architectures. Second, most popular heterogeneous computing architectures have only one kind of accelerator, e.g., a CPU plus a GPU accelerator or a CPU plus an FPGA accelerator, and heterogeneous computing systems with multiple kinds of accelerators, e.g., CPU/GPU/FPGA or CPU/FPGA/NPU

Manuscript received 1 December 2023; revised 20 February 2024 and 15 April 2024; accepted 30 April 2024. Date of publication 8 May 2024; date of current version 8 August 2024. This work was supported by the Ministry of Education of China (MOE) and Intel Industry-University Cooperation Collaborative Education Project under Grant 220900015192812 and Grant 220800015230202. *(Corresponding author: Mengya Zhang.)*

Xing Liu is with the Hubei Key Laboratory of Transport Internet of Things, Wuhan University of Technology, Wuhan 430070, China (e-mail: liu.xing@whut.edu.cn).

Wenxing Xu and Qing Wang are with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China (e-mail: xwx999@whut.edu.cn; 1442459023@whut.edu.cn).

Mengya Zhang is with the School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430070, China (e-mail: zhang_mengya@whut.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3397649

computing architectures, have not been investigated in depth.

To address the above challenges, this article proposes a CPU/FPGA/NPU heterogeneous computing architecture along with a heterogeneous programming model to achieve energy-efficient computing acceleration for computationally intensive UAV tasks. First, we select a set of representative computationally intensive UAV tasks and use these tasks as examples to investigate how to use heterogeneous computing techniques to accelerate UAVs. Then, we design CPU/FPGA and CPU/NPU heterogeneous computing architectures dedicated to these representative UAV tasks and calculate the completion time and energy cost of running these tasks on these heterogeneous computing systems. By analyzing the calculation results, we find that some tasks are appropriate for running on FPGA accelerators, whereas others are more suitable for running on NPU accelerators. Therefore, we further build a CPU/FPGA/NPU heterogeneous computing architecture to achieve better computing efficiency by assigning each UAV task to the computing core, which is the most appropriate for execution. In this way, both the completion time and energy cost of the UAV tasks can decrease significantly.

The contributions of this article are as follows.

1) We designed NPU accelerators for computationally intensive UAV tasks and compared their accelerating performance with that of FPGA accelerators. Since the NPU is an emerging processor, the CPU/NPU accelerating work presented in this article can be of great significance to heterogeneous computing research.
2) We propose a CPU/FPGA/NPU heterogeneous architecture and a heterogeneous programming model for the purpose of achieving in-depth optimization of both the completion time and energy cost of computationally intensive UAV tasks. Currently, there are many studies on CPU/GPU and CPU/FPGA heterogeneous computing, and there are some studies about CPU/NPU heterogeneous computing. However, for CPU/FPGA/NPU heterogeneous computing, the related research is limited. Therefore, our research in this area is innovative and can contribute to this field.

The remainder of this article is structured as follows: in Section II, related work on UAV computing acceleration is introduced. In Section III, an energy-efficient and real-time system architecture for UAVs is described. In Section IV, the representative computationally intensive UAV tasks used to perform computational acceleration are presented. In Section V, the optimization strategies for designing energy-efficient FPGA accelerators for UAV tasks are discussed. In Section VI, the optimization strategies for designing energy-efficient NPU accelerators for UAV tasks are investigated. In Section VII, the heterogeneous CPU/FPGA/NPU computing architecture is proposed, and the CPU/FPGA/NPU heterogeneous programming model is also explored. In Section VIII, experiments are conducted to evaluate the completion time and energy cost of computationally intensive UAV tasks on different heterogeneous computing architectures. Finally, in Section IX, the conclusion is given.

II. Related Work

Currently, the CPU/GPU and CPU/FPGA heterogeneous computing architectures have been popularly applied to accelerate the computationally intensive tasks on UAVs.

A. Computing Acceleration to UAVs Based on CPU/GPU

GPUs commonly have a large number of computing units, ranging from thousands to tens of thousands. This architectural feature makes them demonstrate high-computing performance in parallel processing. In the past few years, many works have applied GPUs to accelerate UAV tasks. Chakravarthy et al. [11] proposed a novel deep learning approach for performing robust semantic segmentation of aerial scenes captured by UAVs. They use the power-constrained UAVs to collect data, and then offload the computationally intensive tasks to a GPU cloud server to achieve computing acceleration. Jaiswal and Kumar [12] investigated the GPU parallel implementation strategies for accelerating the algorithms like feature detection, feature matching, image transformation, frame differencing, morphological processing and connected component labeling for the purpose of achieving real-time detection to the moving objects in UAV-sourced videos. Wang et al. [13] presented a new forward-exploration and backward-navigation algorithm named Matrix Alignment Dijkstra to pilot UAVs, and use GPUs to accelerate the computationally intensive matrix alignment operations of this algorithm. Mousa and Hussein [14] proposed an efficient UAV-based mobile-edge computing mechanism, and use GPUs to accelerate the particle swarm optimization (PSO) algorithm to balance the cluster sizes and identify the shortest path along these clusters while minimizing the UAV flying time and energy consumption. Huang et al. [15] presented a UAV aided lightweight target information collection and detection approach, and they use GPU to accelerate the target detection operations. The results show it achieves a detection accuracy of 89.5% and 71FPS detection speed.

GPU can effectively accelerate many computationally intensive tasks by executing them with high-computational parallelism. However, the power consumption of GPU is high, which reduces the lifetime of the resource-constrained UAVs [16]. Therefore, many research efforts have strived to use the low-power and programmable FPGAs to achieve energy-efficient computing acceleration to the UAV tasks.

B. Computing Acceleration to UAVs Based on CPU/FPGA

Suh et al. [17] proposed an algorithm-hardware co-optimization mechanism based on FPGA to achieve energy-efficient object detection to the UAV applications, such as surveillance, defense, and multidrone self-localization and formation control. The evaluation results show that this mechanism can achieve high-energy efficiency and high throughput. Zhang et al. [18] presented an FPGA-based convolutional neural network (CNN) accelerator named FitNN. By reducing the data movements of intermediate results, both the speed and power efficiency of CNN inference are improved. Sadhu et al. [19] proposed a novel architecture based on deep convolutional network, long short-term memory neural networks and FPGA hardware accelerator, for the purpose of detecting and classifying UAV misoperation. The result show that FPGA accelerator can achieve a speedup of 40x for detection, while consuming only half the amount of power compared with onboard NVIDIA Jetson TX2 GPU. He et al. [20] proposed a new UAV returning framework without global positioning system (GPS) based on the improved Kanade–Lucas–Tomasi (KLT) feature tracker, and they use FPGA to accelerate the high-computational complexity of KLT. The results show that the processing speed can achieve 60 frames per second. Amanatiadis et al. [21] presented a moment-based blob detection approach to achieve real-time surveillance detection for medium-altitude long-endurance UAVs, and they use FPGAs' parallelization to achieve real-time performances and high-energy efficiency. Li et al. [22] proposed an improved YOLOv7 for small object detection in UAV image scenarios. They implement YOLOv7 on FPGA by using Vitis-AI development tools, and the results show that the FPGA implementation improves the energy efficiency by 12 times compared to the GPU.

C. Computing Acceleration to UAVs Based on CPU/NPU

In recent years, NPU has been gradually used in industry fields, yet the research work on NPU in academia is still insufficient. Tan and Cao [23] proposed a framework called FastVA, which supports deep learning video analytics through edge processing and Huawei's NPU on Mate 10 Pro smartphone, for the purpose of decreasing delay and energy consumption of mobile applications. Wang et al. [24] focused on the neural network architecture design for NPUs. They propose a novel architecture named Serial and Parallel Group Network (SPGNet) aiming to make the network structure compact. Consequently, the network can be more friendly to the existing NPU hardwares.

There are also some other research works about NPUs. However, these works are not carried out on the real NPUs. Instead, they use the programmable FPGA to design NPU accelerators to accelerate the execution of some tasks with high-computational complexity [25], [26], [27].

D. Computing Acceleration Based on Heterogeneous Systems With Multiple Accelerators

Since different accelerators are suitable to be used in different contexts, a heterogeneous computing system with multiple accelerators can become context aware to different application scenarios, and therefore achieve better computing efficiency. However, there are currently not many heterogeneous computing products which have multiple accelerators, due to the difficulties in manufacture and heterogeneous programming.

NVIDIA Jetson AGX Xavier (NJAX) is a commercial embedded device with heterogeneous accelerators. It is equipped with one ARMv8 CPU, one Volta GPU, and two NVIDIA deep learning accelerators. In the past years, NJAX has been applied by some researchers to conduct the heterogeneous computing research. Kim and Ha [28] proposed a novel energy-aware mapping methodology for multiple deep learning applications on NJAX aiming to minimize the energy consumption while satisfying the real-time constraints of deep learning applications. Jeong et al. [29] proposed a parallelization methodology to maximize the throughput of a single deep learning application by using both GPU and NPU on NJAX through various types of parallelism on TensorRT.

Inspired by the big.LITTLE computing paradigm, Spantidi et al. [30] implement heterogeneous DNN accelerators (HDAs) which consisting of 8-bit NPUs together with lower precision bit-width NPUs to increase the overall throughput, while reducing the energy consumption during NN inference.

Overall, the research work on heterogeneous computing systems with multiple kinds of accelerators, especially with NPU cores, has not received enough research attention.

III. Energy-Efficient and Real-Time System Architecture for UAV

This section presents an energy-efficient and real-time heterogeneous system architecture for UAVs, which is depicted in Fig. 1. This architecture integrates a set of enabling technologies, such as heterogeneous computing, heterogeneous task scheduling, Kubernetes container management, and edge-cloud collaborative computing. Based on this architecture, both the computing speed and energy cost of the UAVs can be optimized. In this article, we focus on presenting the work of heterogeneous computing system in the low-level layer of this architecture.

A. CPU/FPGA/NPU Heterogeneous Computing

Different types of processors have different hardware architectures and are therefore good at executing different kinds of UAV tasks. To improve the computing efficiency of UAVs, heterogeneous processors, such as CPUs, FPGAs, and NPUs, can be integrated on UAVs. At run-time, each task can be assigned to the most suitable cores for execution. In this way, the UAV tasks can be executed more efficiently, resulting in less completion time and lower energy cost.

B. Heterogeneous Scheduling Algorithm

Dedicated to the CPU/FPGA/NPU heterogeneous system, a heterogeneous task scheduling algorithm needs to be explored. This algorithm assigns each task to the cores which are the most suitable to run this task, thereby improving the computing efficiency of the different tasks.

To search for an optimal task scheduling solution, the task model, energy model and time model of the UAV tasks in the heterogeneous system need to be built. Then, the scheduling optimization problem can be formulated. In most cases, the optimization objective is to minimize the energy consumption while satisfying the real-time constraints. Following that, a scheduling algorithm to search for an optimal solution needs to be studied.

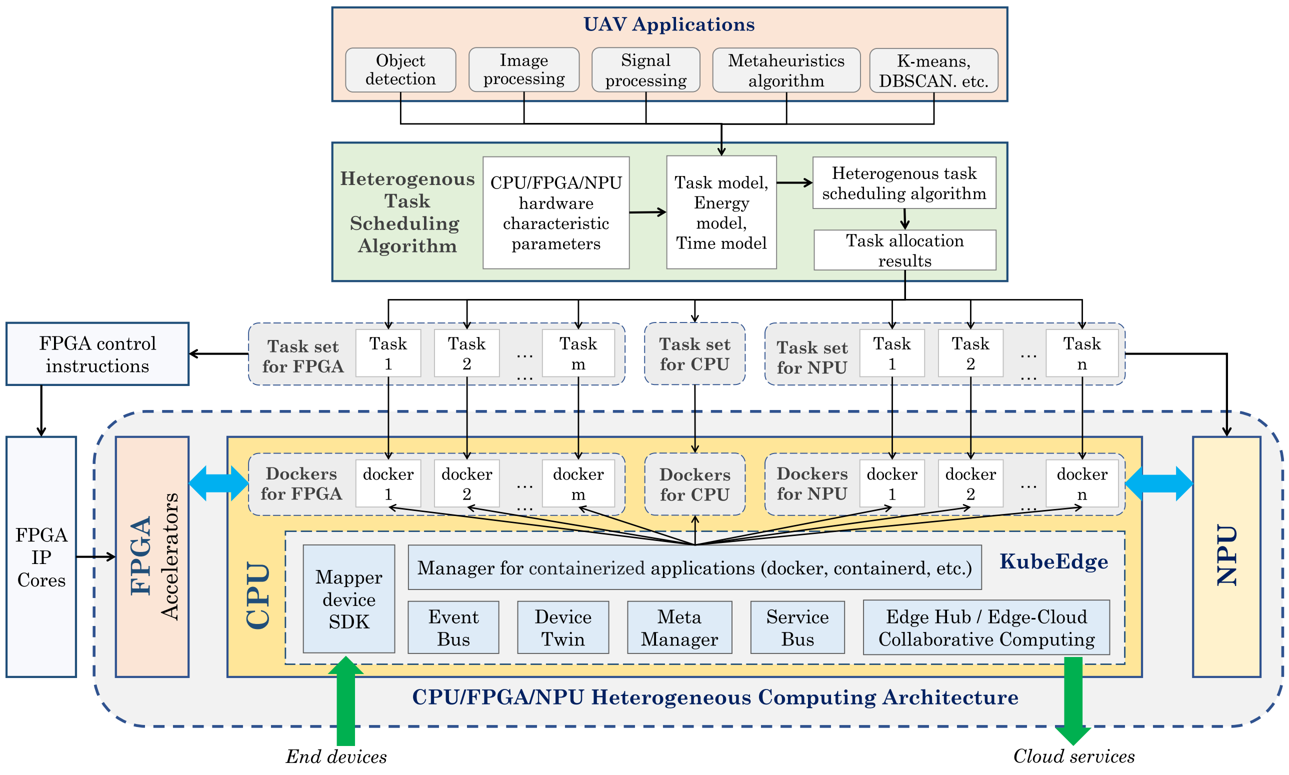


Fig. 1. Energy-efficient heterogeneous computing architecture for UAV system.

C. Dynamic FPGA Accelerators

FPGA accelerators are designed to support dynamic reconfiguration at runtime to strike the tradeoff between performance and versatility.

On the one hand, FPGA accelerator can achieve high-computing performance since its hardware is programmable and can be converted into a specific architecture dedicated to a given task. On the other hand, the specific architecture reduces the versatility of FPGA accelerators. To strike a tradeoff between performance and versatility and make FPGA accelerators adapted to a variety of application contexts, a controller is implemented inside the FPGA accelerators. During the run-time, some instructions can be injected into this controller to make the circuit components inside FPGAs to operate in different ways. With this mechanism, the FPGA accelerators can become more versatile.

In addition, the dynamic reprogramming technology can also be implemented for FPGAs to further enhance its versatility. In the off-chip storage system, a variety of FPGA IP cores are stored, and these IP cores can be reprogrammed into FPGA to rebuild the FPGA's functionality.

D. Edge-Cloud Collaborative Computing Based on KubeEdge

On the CPU cores of UAVs, the KubeEdge edge computing system is deployed to extend the native containerized application orchestration and device management to the hosts at edge [31]. Based on KubeEdge, the UAVs can have the capabilities of networking, edge-cloud collaborative computing as well as containerized application deployment.

1) Basic Edge Computing Services: KubeEdge enables UAVs to collect data from other end devices as well as send commands to control other end devices. Based on KubeEdge, the UAVs can cooperate with each other more conveniently. In addition, to enable UAVs with new protocols connect to KubeEdge in an easy way, we develop a KubeEdge device management SDK [32]. This SDK builds the common parts of the functional components of different protocols into libraries, and then inject configuration information into KubeEdge to make the SDK dynamically adapt to different protocols. With this SDK, the developers only need to implement the universal interfaces when connecting a new protocol to KubeEdge, and this greatly simplifies the development complexity of UAVs.

2) Manager for Containerized Applications: Since KubeEdge is built upon Kubernetes, the tasks assigned to CPUs, NPUs and FPGAs on the heterogeneous UAVs can be built into containers. At runtime, KubeEdge uses Kubernetes to manage these containers to realize automating deployment, scaling, rollouts and rollbacks, service discovery, load balancing, self-healing and storage orchestration. These functionalities effectively improve the reliability and scalability of UAV system. Considering the computing performance and energy resources of UAV are constrained, we substitute the original resource-intensive Kubernetes in KubeEdge with the lightweight Kubernetes, also known as k3s, when deploying KubeEdge on UAVs.

3) Edge-Cloud Collaborative Computing: KubeEdge can also provide edge-cloud collaborative computing service for UAVs [33]. On the edge side, KubeEdge has an edge hub. On the cloud side, it provides the cloud hub, edge controller and also device controller. Based on these components, the edge-side status change can be reported to the cloud while the cloud-side resource updates can also be synchronized to the edge. The above features enable KubeEdge to provide edge-cloud synergy computing capabilities, such as edge-cloud joint inference, edge-cloud incremental training, edge-cloud federated learning and edge-cloud lifelong learning for the UAV devices.

IV. Representative Computationally Intensive UAV Tasks for Computing Acceleration

In many UAV applications, a large number of computationally intensive tasks need to be executed [34], [35], and these tasks commonly have long computation times and high-energy costs. Therefore, it is important to accelerate these tasks to optimize their time and energy costs. As a research effort, accelerating all computationally intensive UAV tasks is impossible. Thus, selecting representative UAV tasks and then using them as examples to investigate methods to accelerate UAVs is necessary.

The popular computationally intensive UAV tasks include image processing tasks (image segmentation, image feature extraction, etc.), AI-driven tasks (image classification, object detection, etc.), and signal processing tasks (signal denoising, signal feature extraction, etc.). Thus, we select six image processing algorithms (mean filter, Gaussian filter, median filter, bilateral filter, Laplacian filter, Otsu), one image classification algorithm (MobileNet), one object detection algorithm (YOLO), and one signal processing algorithm (FFT), as representative tasks to investigate how to use heterogeneous computing systems, such as CPU/FPGA and CPU/NPU, to achieve energy-efficient computing acceleration for UAV tasks.

In the following sections, we first design FPGA and NPU accelerators for the above algorithms to accelerate the computations of these algorithms. Then, we compare the completion time and energy cost of these algorithms on the FPGA and NPU accelerators, respectively. Based on the comparison results, we can discover which types of tasks are most suitable for which kinds of accelerators, and this finding can be significant for guiding the architectural design of future heterogeneous systems.

V. Energy-Efficient Computing Acceleration to UAV With CPU/FPGA Architecture

In this section, we present the work on how to design FPGA accelerators for the representative computationally intensive UAV tasks to realize energy-efficient computing acceleration to these tasks.

A. Architectural Design of FPGA Accelerators

Assuming the mask size of the image filtering algorithms is 3 × 3, then nine buffers need to be used to construct a 3 × 3 matrix. To increase the computational parallelism, the matrix multiplication is operated row by row, and the matrix data can be updated by shifting the data in each row of buffers. In this way, the matrix multiplication of the image filtering algorithms can be operated efficiently. According to the above computing concept, the hardware architectures of various filtering algorithms are designed, e.g., in Fig. 2, the FPGA architecture for performing bilateral filtering operation is depicted.

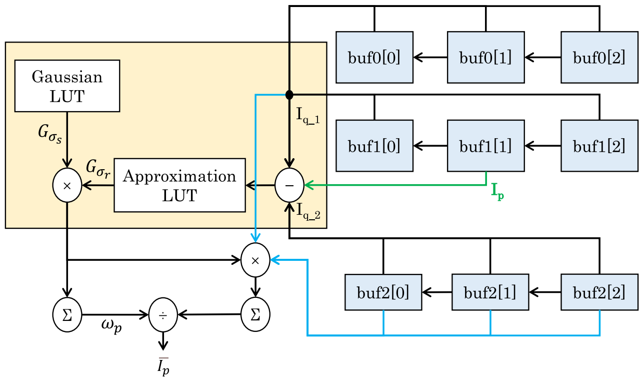


Fig. 2. FPGA architecture for performing bilateral filter.

The architectures of FPGA accelerators for mobileNet and FFT have been presented in our previous work [36], [37].

B. Optimization Strategies for Designing FPGA Accelerators

The optimization strategies used for designing FPGA accelerators include parallelism, pipelining, tiling, data multiplexing, algorithm optimization, data quantization, etc.

1) Computing Parallelism: The architecture of FPGA accelerator needs to increase the computing parallelism to improve the computing speed, and this is a commonly used computing acceleration strategy that has been used in all FPGA accelerators designed in this article, e.g., in Fig. 2, the matrix multiplication is performed in parallel row by row, and the multiplication of each row is also performed in parallel through multiple multipliers.

2) Pipelining: Some operations cannot be executed in parallel due to the data dependencies. For these operations, the pipelining can be used to increase the computing performance. For example, the sum of probability values in Otsu algorithm can be calculated as follows:

$$p[i] \leftarrow pCnt[i]/(\text{width} \times \text{len}) \quad (1)$$

$$tp[i] \leftarrow tp[i-1] + p[i] \quad (2)$$

$$M_k[i] \leftarrow M_k[i-1] + i \times p[i] \quad (3)$$

in which *width* and *len* represent the length and width of image pixels respectively, $pCnt[i]$ represents the number of pixels for different grayscale values, $p[i]$ is the probability value, and $tp[i-1]$ is the intermediate result of probability value. Since the subsequent operation needs to use the output of previous operation, the computing process cannot be performed in parallel. In this case, the pipelining mechanism can be used to improve the computing speed. In Fig. 3, a pipelined computing architecture is designed to calculate the probability of grayscale values.

3) Data Multiplexing: The memory read and write operations usually have high-operating cost, which makes it necessary to decrease the number of memory accesses. To

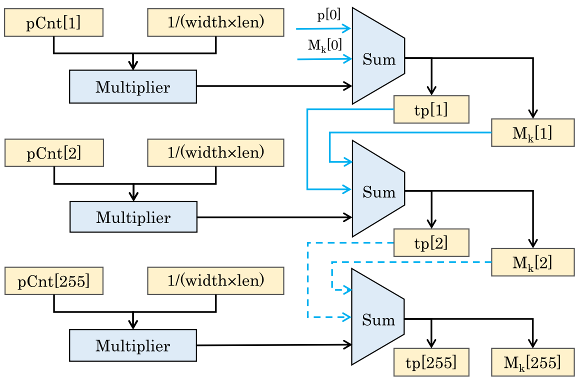


Fig. 3. Architecture to calculate probability value in Otsu.

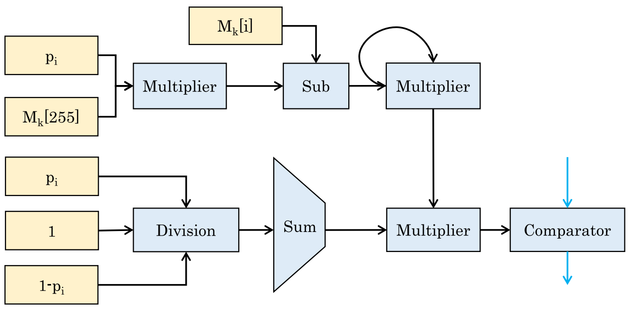


Fig. 4. Architecture to calculate thresholding value in Otsu.

achieve the above objective, the data multiplexing technology is used for designing the FPGA accelerators, e.g., when performing image filtering on FPGAs, nine buffers are used to store the matrix data, and the update to the matrix can be performed by shifting the data in the buffers. In this way, the duplicate data in two matrices do not need to be read repeatedly from the memory system, which can significantly reduce the memory operation cost.

4) Algorithm Optimization: In some cases, the operating process of the algorithm needs to be adjusted to make the algorithm more suitable for implementation on FPGA. In the Otsu algorithm, the threshold value can be calculated as follows:

$$\text{threshold} = \frac{(m_k - p_i \times mG)^2}{p_i \times (1 - p_i)}. \tag{4}$$

Since this process involves the division operation with floating point numbers, it has high-computational overhead. To decrease the computing complexity, we convert the division operation into multiplication operation with the following conversion:

$$\frac{(m_k - p_i \times mG)^2}{p_i \times (1 - p_i)} = (m_k - p_i \times mG)^2 \times \left(\frac{1}{p_i} + \frac{1}{1 - p_i}\right). \tag{5}$$

Moreover, we use the quantization mechanism to convert the floating number p_1 into an integer for operation. In this way, the floating-point division operations can be avoided, and this not only reduces the complexity of the hardware architecture but also improves the computing efficiency of Otsu algorithm. In Fig. 4, the architecture to calculate the threshold is depicted.

5) Data Quantization: Data quantization can reduce the computing complexity and increase the computing performance of FPGA accelerators. When designing FPGA accelerator for bilateral filter, the value G_{σ_r}, as is shown in (6), needs to be calculated. Since this calculation process involves exponential and floating-point number operations, the computing complexity is high. To reduce computational complexity, the data quantization approach is used. First, the value of G_{σ_r} is calculated in advance and sampled at 255 equal intervals. Then, the sampled values are stored into the FPGA's lookup table (LUT). In this way, FPGA no longer needs to calculate G_{σ_r} during the run-time. Instead, it only needs to look up the value directly from the LUT, as is depicted in Fig. 2, and this way can significantly improve the computing speed of FPGA accelerator. Although data quantization can reduce the result accuracy to a certain extent, this loss of accuracy can be controlled to an acceptable range in most cases.

$$G_{\sigma_r} = e^{-\frac{x^2}{2\sigma_r^2}}, x \in [0, 255]. \tag{6}$$

VI. Energy-Efficient Computing Acceleration to UAV With CPU/NPU Architecture

In Section V, we have discussed the way to design FPGA accelerators for the representative computationally intensive UAV tasks. In this section, we further investigate the way to design NPU accelerators for the representative tasks, and we use Huawei's Ascend NPU as example for this discussion.

A. Computing Framework of Huawei's Ascend NPU

The computing framework of Huawei's Ascend AI is depicted in Fig. 5.

In the heterogeneous computing framework layer, the Huawei Compute Architecture for Neural Networks (CANN) is provided [38]. CANN is a heterogeneous computing architecture for AI scenarios, and it provides a set of multilayer programming interfaces which can help developers to quickly build the AI applications based on the Ascend NPU. In the AI framework layer, the Huawei MindSpore framework is designed [39]. MindSpore is a new open source deep learning training/inference framework, and it can provide development experience with friendly design and efficient execution for AI developers.

B. Acceleration of Representative Tasks Base on CPU/NPU

To explore the strengths and weaknesses of NPU processors, we also implement all the representative tasks on Huawei's Ascend NPU.

1) Cube Unit and Vector Unit: The architecture of AI core in Huawei's Ascend NPU is depicted in Fig. 6. It contains three kinds of computing units: 1) cube unit; 2) vector unit; and 3) scalar unit. The cube unit is designed to perform 3-D convolutional operations, while the vector unit can be used to perform activation and pooling functions.

As the image filtering operation is operated by 2-D matrix multiplication, rather than 3-D multichannel matrix multiplication, it is commonly not suitable to use the cube unit to

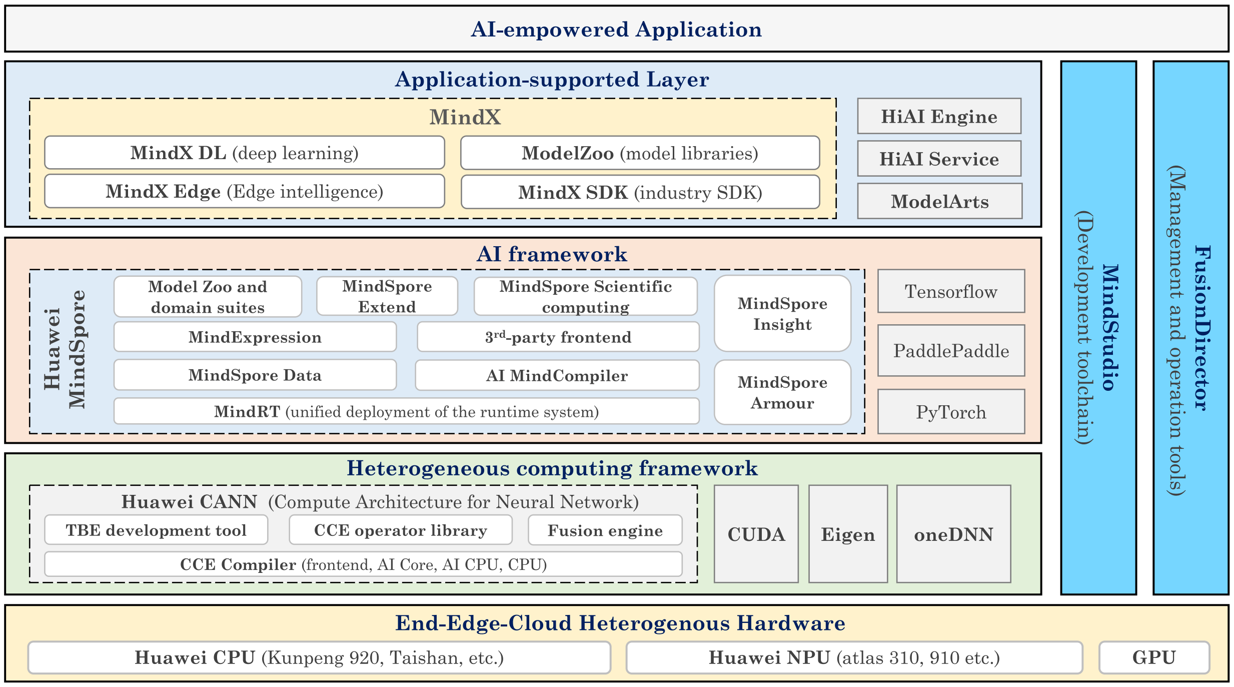

AI-empowered Application
Application-supported Layer
MindX
MindX DL (deep learning)
ModelZoo (model libraries)
MindX Edge (Edge intelligence)
MindX SDK (industry SDK)
HiAI Engine
HiAI Service
ModelArts
AI framework
Huawei MindSpore
Model Zoo and domain suites
MindSpore Extend
MindSpore Scientific computing
MindSpore Insight
MindExpression
3rd-party frontend
MindSpore Data
AI MindCompiler
MindSpore Armour
MindRT (unified deployment of the runtime system)
Tensorflow
PaddlePaddle
PyTorch
Heterogeneous computing framework
Huawei CANN (Compute Architecture for Neural Network)
TBE development tool
CCE operator library
Fusion engine
CCE Compiler (frontend, AI Core, AI CPU, CPU)
CUDA
Eigen
oneDNN
MindStudio (Development toolchain)
FusionDirector (Management and operation tools)
End-Edge-Cloud Heterogenous Hardware
Huawei CPU (Kunpeng 920, Taishan, etc.)
Huawei NPU (atlas 310, 910 etc.)
GPU

Fig. 5. Computing framework of Huawei's Ascend AI.

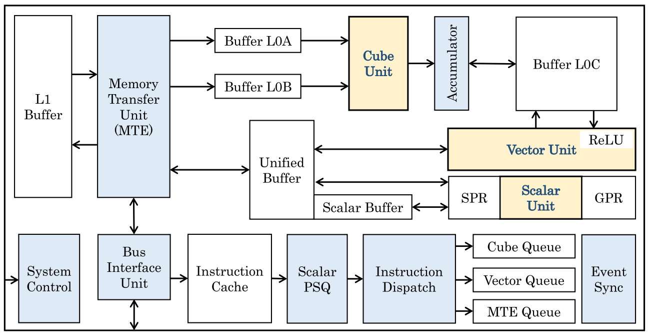


Fig. 6. Architecture of AI core in Huawei's Ascend NPU.

implement the image filtering algorithms. Therefore, the vector units is used to achieve this function on NPU. In Fig. 7, the mechanism of using vector computing unit of NPU to perform image filter operations is depicted. The matrix multiplication is implemented in parallel by row-wise vector multiplication. By means of the data multiplexing and buffer shifting technology, the number of memory accesses during the filtering process can be greatly reduced, thereby efficiently implementing the filtering operations on NPU.

2) How to Use Low-Precision Components to Implement Complex Operations: By using low-precision FP16 to perform vector computing, Ascend NPU can provide powerful computing power. However, low-precision calculations are prone to cause overflow problem. Taking the (6) as example, if the value of $2\sigma_r^2$ is less than 1, the value of $(x^2/2\sigma_r^2)$ can be overflowed. To solve this problem, a large value that is prone to be overflowed can be decomposed into multiple small values that are nonoverflowed. Specifically, the calculation process of G_{σ_r} in (6) can be converted into the ways as follows:

$$G_{\sigma_r} = e^{-\frac{x^2}{2\sigma_r^2}} = e^{-\frac{128^2\times a}{2\sigma_r^2}} \times e^{-\frac{128\times b}{2\sigma_r^2}} \times e^{-\frac{c}{2\sigma_r^2}} \tag{7}$$

in which $(abc)_{128}$ is equal to x^2. In this way, the overflow problem in low-precision calculations can be avoided.

3) Task Allocation Between CPU and NPU: In the CPU/NPU heterogeneous system, only the tasks suitable for NPU need to be offloaded to NPU, whereas the other tasks remains to be executed on the host CPU.

Use the Otsu algorithm as example, only the operations of calculating threshold and performing binarization are implemented on NPU, whereas the other operations, such as statistical operations and probability calculations, are realized on the host CPU, and this is because these two kinds of operations are not suitable to be implemented on NPU due to the following reasons: First, the infrastructure of NPUs is suitable for executing the highly parallel tasks, yet the statistical calculation operation does not meet this requirement. Second, the statistical operations involve the accumulation of different intermediate results, and it is inefficient to use NPU to perform this operation since it does not have the specific component to perform this function. Third, when accumulating the intermediate results on NPU, frequent data movement are operated, which can increase the computational overhead.

Due to the above reasons, the different tasks of Otsu algorithm are distributed to CPU and NPU respectively. In Fig. 8, the task allocation mechanism of Otsu algorithm on the CPU/NPU heterogeneous system is depicted. First, CPU calculates the statistical value and probability value through multiple iterations. Then, it transmits these values to NPU. Later, NPU works out the threshold value and performs

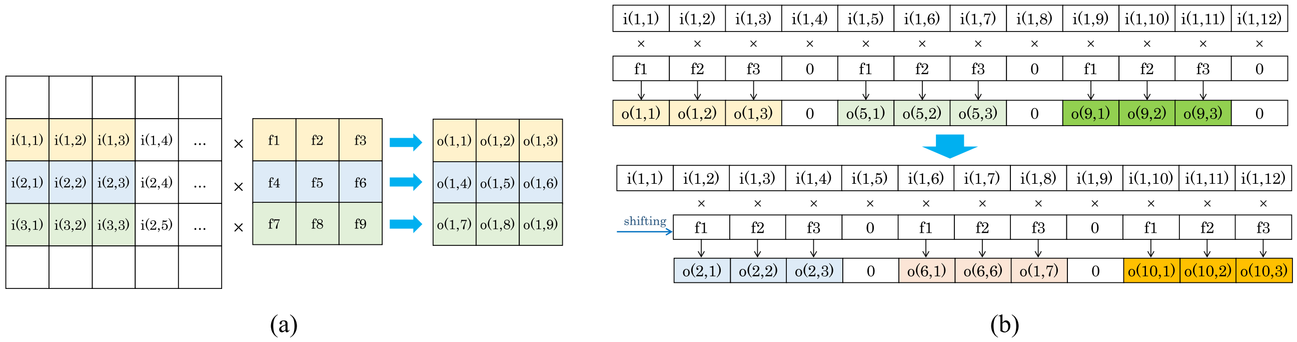


Fig. 7. Implementation of image filtering operations on Huawei's Ascend NPU. (a) Image filtering operations. (b) Implementation of image filtering operations on NPU.

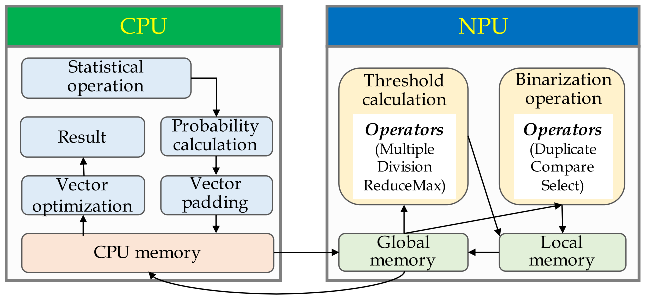


Fig. 8. Task allocation between CPU core and NPU core.

binarization operations to generate the final results. Finally, NPU transmits the results back to CPU.

4) Not All Tasks Are Suitable for Implementing on NPU: NPU shows good performance in parallel computing, and is appropriate to perform acceleration for vector operations and matrix operations. However, for the tasks that are hard to be processed in parallel by using vector or matrix operations, they are not suitable to be accelerated by NPU. Take FFT as an example, it does not show good performance on NPU due to the following reasons: 1) the data that needs to be processed cannot be accessed continuously, resulting in frequent memory access operations; 2) a large number of intermediate results need to be accumulated, which can lead to large intercore data exchange cost in NPU; and 3) the operators of multiplication, addition and subtraction require to split the real part and imaginary part of the complex, which increases the cost of on-chip memory operations.

5) NPU Is Specially Designed for AI Models: The hardware architecture of NPU is specially designed for AI models. The cube unit of the AI core in NPU shows strong performance when performing convolutional computing tasks, while the vector unit in NPU is good at performing activation and pooling operations. In addition, many AI frameworks have been designed for NPU, e.g., the Huawei Ascend NPU has supported the AI frameworks, such as MindSpore, Tensorflow, and PyTorch. Based on the above software and hardware infrastructure, it is convenient to deploy an AI model on NPU. In this article, we implement the mobilenet and YOLO on Ascend NPU by using the Huawei's mindspore.

VII. Energy-Efficient Computing Acceleration to UAV With CPU/FPGA/NPU Architecture

The experimental results in Section VIII show that different tasks are suitable to run on different cores. Therefore, in this section, we built a CPU/FPGA/NPU heterogeneous system and assigned each UAV task to the most suitable core for execution. In this way, both the time and energy cost of UAV tasks can be further optimized. In this section, we present the computing architecture and heterogeneous programming model for the CPU/FPGA/NPU heterogeneous system.

A. CPU/FPGA/NPU Heterogeneous Computing Architecture

CPUs, FPGAs and NPUs all have different computing architectures and are therefore suitable for performing different types of tasks. To take advantage of different processors, the CPU/FPGA/NPU heterogeneous computing system is built. At run-time, the CPUs can undertake the tasks which have simple computational complexity but complex control logic, e.g., the conditional branch, data initialization and communication control. The NPUs, which is specially designed for AI models, can be used to run the AI tasks, such as CNNs and recurrent neural networks (RNNs). As for FPGAs, due to their hardware programmable features, they can be used to build the specialized accelerators dedicated to the computationally intensive algorithms in addition to AI tasks, e.g., the FFT operations.

With the above heterogeneous computing mechanism, the tasks can be executed more efficiently, thereby reducing both the completion time and energy consumption.

B. Programming for CPU/FPGA/NPU Heterogeneous System

The CPU/FPGA/NPU heterogeneous computing architecture can adapt to different application contexts, thus improving the computing efficiency. However, the programming on the CPU/FPGA/NPU heterogeneous system is challenging since CPUs, FPGAs and NPUs all have different programming languages and development tools, as is depicted in Fig. 9, and it is difficult to unify the different tools and languages.

To solve the above problem, we proposed a user-friendly CPU/FPGA/NPU heterogeneous programming model by improving the Intel's oneAPI. oneAPI is an open,

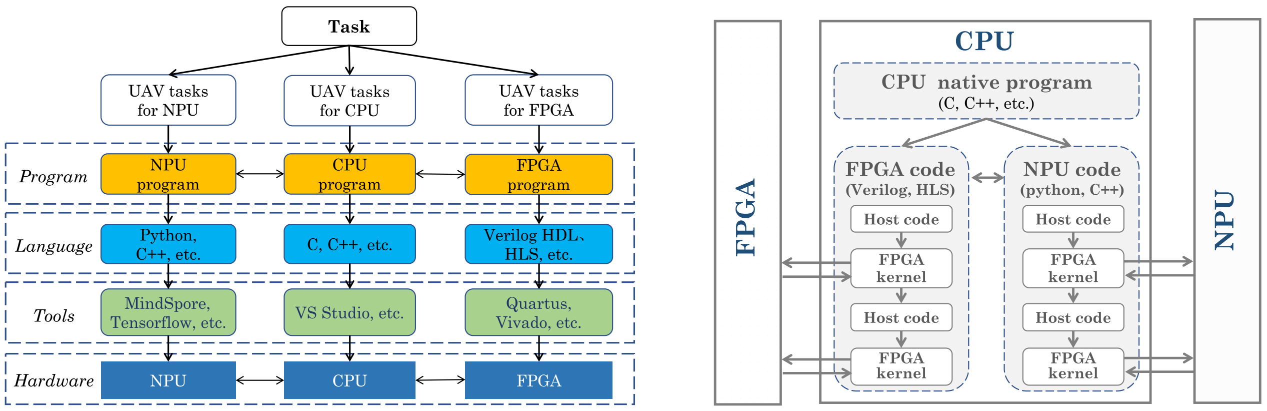


Fig. 9. Traditional way to program on the CPU/FPGA/NPU heterogeneous computing system.

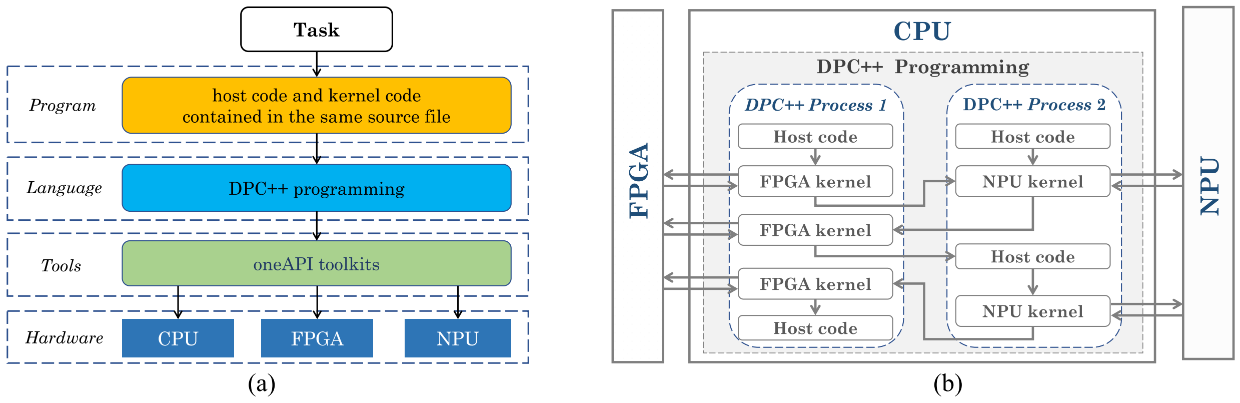


Fig. 10. Programming model for CPU/FPGA/NPU heterogeneous computing system. (a) Intel oneAPI programming model. (b) Unified programming model for CPU/FPGA/NPU system based on oneAPI.

cross-industry, standards-based, unified, multiarchitecture, multivendor programming model that delivers a common developer experience across accelerator architectures, as is depicted in Fig. 10(a). It enables the developers to use the consistent programming language DPC++, which is a cross-architecture language built upon C++ and SYCL standards, to develop on the different heterogeneous computing platforms, such as CPU/GPU, CPU/FPGA, and CPU/NPU [40]. Currently, oneAPI supports many mainstream accelerators, such as Nivida's GPUs, Intel's FPGA, and also Huawei's NPU [41].

However, oneAPI can only support the programming on the heterogeneous system with one kind of accelerators, e.g., CPU/GPU or CPU/FPGA, rather than the heterogeneous system with multiple types of accelerators, e.g., CPU/GPU/FPGA or CPU/FPGA/NPU. To solve this problem, we improved oneAPI programming model by creating two DPC++ processes, and use one process to manage the CPU/FPGA heterogeneous system, while the other to manage the CPU/NPU heterogeneous system. At runtime, two processes interact with each other to perform data exchange and computing collaboration, as is depicted in Fig. 10(b). In this way, the developers can still use the unified DPC++ language to program on the CPU/FPGA/NPU heterogeneous system, and the programming complexity is significantly decreased.

VIII. Experimental Results

In this section, experiments are conducted to evaluate the completion time and energy cost of running all the representative computationally intensive UAV tasks on different heterogeneous systems, such as CPU/FPGA, CPU/NPU, and CPU/GPU.

A. Experimental Environment Setup

The Intel i5-9300H CPU, Nivida GTX1650 GPU, Intel Cyclone 10GX FPGA and Huawei Ascend 310p NPU are used as the experimental platforms. The CPU has 8 cores working at the frequency 2.4 GHz, and the operating system is Ubuntu 20.04 LTS. The GPU has 896 CUDA cores, 4-GB memory and the maximum power is 75 W. The FPGA has 104000 logic units, 8.439-Mb RAM and 125 DSP. The NPU has eight CPU cores working at the frequency 1.9 GHz, and eight DaVinci AI Cores with 140 TOPS@INT8 and 70 TOPS@FP16.

The GPU is developed by CUDA. The FPGA is developed by Quartus, and the programming language is Verilog HDL. The NPU is developed by CANN and mindSpore, and the programming language is python and Ascend C.

The power consumption of CPU, GPU, FPGA, and NPU is obtained, respectively, by using HWMonitor software, NVIDIA nvidia-smi command, FPGA synthesis report and npu-smi command.

TABLE I
COMPLETION TIME OF RUNNING REPRESENTATIVE TASKS ON DIFFERENT HETEROGENEOUS SYSTEMS

Tasks	**Algorithms**	**Parameters**	**Completion Time of Different Tasks (ms)**			
		(image size or data size)	CPU	CPU/GPU	CPU/FPGA	CPU/NPU
Image Processing	Mean Filter	256 × 256	48.1	**0.5907**	0.6580	9.7
		1024 × 1024	1585.8	**3.6156**	10.4858	365.1
	Gaussian Filter	256 × 256	20	0.6474	**0.6295**	12
		1024 × 1024	856.2	**3.9207**	10.4856	234.5
	Laplacian Filter	256 × 256	43.1	**0.477**	0.5189	8.3
		1024 × 1024	656.2	**3.9641**	9.4727	109.4
	Median Filter	256 × 256	140.3	121.329	**0.7294**	52.7
		1024 × 1024	2943.3	2257.16	**10.7435**	784
	Bilateral Filter	256 × 256	882.2	94.156	**1.0154**	192.2
		1024 × 1024	9125.2	1013.31	**11.0825**	1401.2
	Otsu	256 × 256	35.3084	24.1283	**2.57**	5.9
		1024 × 1024	361.293	270.537	**48.14**	163.1
AI models	YOLO v5s	650 × 433	232.1	42.1	102.4	**27**
	MobileNet	810 × 1080	89.16	28	37.3	**18.41**
Signal Processing	FFT	4000000	189.46	**16.215**	40.234	210.32

B. Completion Time of Representative Tasks on Different Heterogeneous Systems

The completion time of running all the representative tasks on different heterogeneous computing systems is depicted in Table I. We can obtain the following findings from the results.

1) The mean filter and Laplacian filter algorithms run fastest on the GPU, followed by the FPGA, and slowest on the NPU. This is because these filtering algorithms mainly perform linear operations, and the GPU can make full use of its parallel computing capability to achieve significant acceleration of these operations. Although the cube unit of the NPU also has high-parallel computing capability, it is not suitable for filtering operations since these operations mainly involve 2-D rather than 3-D matrix calculations. Consequently, it is more suitable to use the vector unit rather than the cube unit to process the filtering tasks on the NPU, which limits the computing performance of the NPU, making its performance in running these algorithms inferior to that of the GPU.
2) The median filter algorithm runs fastest on the FPGA, followed by the NPU, and slowest on the GPU. This is because the median filter involves a sorting operation, which cannot take advantage of the highly parallel computing performance of the GPU. As a result, the GPU cannot achieve good accelerating performance for this algorithm. In contrast to GPUs, FPGAs can adapt to sorting operations efficiently by designing a specific hardware architecture, which improves the FPGA performance in running median filter algorithms. The NPU also achieves better performance than the GPU for executing median filter because the NPU provides efficient operators optimized for performing batch sorting operations.
3) The bilateral filter algorithm runs fastest on the FPGA, followed by the GPU, and slowest on the NPU. This is because this algorithm involves exponential operations, and the FPGA can determine the exponential value in advance and store the sampled exponential value in a lookup table. Thus, complex exponential, multiplication and division operations can be avoided on the FPGA, which can greatly improve FPGA speed.
4) The Otsu algorithm runs fastest on the FPGA, followed by the NPU, and slowest on the GPU. The FPGA can run the Otsu algorithm with high performance since it can provide high-computing parallelism by using tiling technology. Moreover, the hardware can be reprogrammed to build a specific accumulator to efficiently accumulate intermediate values. NPUs and GPUs need to offload parts of Otsu operations to CPUs for execution due to their specific hardware architecture, making their performance inferior to that of FPGAs when executing Otsu operations.
5) AI models, such as YOLO and MobileNet, run fastest on the NPU, followed by the GPU, and slowest on the FPGA. The NPU has high performance for AI models since its hardware architecture is specifically designed for performing AI operations. FPGAs commonly perform worse than GPUs for AI tasks, yet they mostly achieve higher energy efficiency than GPUs.
6) The Gaussian filter algorithm has considerable speed on FPGA and GPU when the image size is 256×256. However, it runs faster on GPU than on FPGA when the image size increases to 1024×1024. This is because the cost of memory operations decreases as a proportion of the entire operation cost when the image size increases.

C. Energy Cost of Representative Tasks on Different Heterogeneous Systems

The power and energy cost of running all the representative tasks on different heterogeneous systems is depicted in Table II.

From the results, it shows that: 1) the power of GPU is commonly higher than that of FPGA and NPU; 2) FPGA has the lowest energy cost when running the algorithms, such as mean filter, Gaussian filter, Laplacian filter, median filter, bilateral filter, Otsu, and FFT; and NPU has the lowest energy cost when running the AI models, such as YOLO and mobileNet; and 3) if an algorithm is not suitable for running on a certain processor, it will result in high-energy cost.

TABLE II
ENERGY COST OF RUNNING REPRESENTATIVE TASKS ON DIFFERENT HETEROGENEOUS SYSTEMS

Tasks	**Algorithms**	**Power (W)**				**Energy (J)**			
		CPU	CPU/GPU	CPU/FPGA	CPU/NPU	CPU	CPU/GPU	CPU/FPGA	CPU/NPU
Image Processing (image size: 1024×1024)	Mean Filter	6.15	9	1.083	0.8	9.75267	0.03254	**0.01136**	0.29208
	Gaussian Filter	8.34	10	1.013	1	7.140708	0.039207	**0.01062**	0.2345
	Laplacian Filter	5.76	9	1.014	0.8	3.779712	0.035677	**0.00961**	0.08752
	Median Filter	7.12	11	1.321	1.4	20.956296	24.82876	**0.0142**	1.0976
	Bilateral Filter	7.31	11	1.487	1	66.705212	11.143	**0.0165**	1.4012
	Otsu	8	9	1.072	1.5	2.89034	2.43	**0.0516**	0.24465
AI Models	YOLO v5s	13.21	17	2.057	2.1	3.066041	0.7157	0.2125	**0.0567**
	MobileNet	12.48	16	1.3	0.7	1.1127168	0.448	0.04849	**0.012887**
Signal Processing	FFT	8.12	11	3.2	0.8	1.5384152	0.178365	**0.1287**	0.784

D. Discussion on the CPU/FPGA/NPU Heterogeneous Computing System

It can be seen from the above results that different tasks can be executed on different processors. Therefore, a heterogeneous computing system with multiple kinds of accelerators can be built for UAVs to improve the computing efficiency of UAV tasks. In our work, we build a heterogeneous CPU/FPGA/NPU architecture for UAVs. We select the NPU because it is specifically designed for running AI tasks. We select the FPGA because it can reprogram its hardware architecture to adapt to different UAV application contexts. A GPU is not used since it commonly has a high-energy cost and is not conducive to the lifetime of UAVs.

We assign each UAV task to the most suitable core for execution during runtime so that each task can be performed with high efficiency. Assume there is an application case as follows: The UAVs need to perform environmental monitoring tasks in real time. UAVs need to first take pictures of the scene and collect vibration signals from the environment. Then, they need to perform Gaussian filter and Otsu operations to preprocess the collected pictures. Next, they ran YOLO to realize target recognition from the preprocessed pictures. In addition, they need to run FFT to process the captured vibration signals and detect anomalies. In this application scenario, UAVs need to execute four types of computationally intensive tasks: 1) the Gaussian filter; 2) Otsu; 3) YOLO; and 4) FFT. To enable these tasks to be processed in real time, each task needs to run on the processor that takes the shortest time. According to the experimental results in Table I, the Gaussian filter and FFT tasks need to be assigned to the GPU, the Otsu operation can be executed on the FPGA, and the YOLO operation should be performed on the NPU. With this heterogeneous computing mechanism, the reaction speed of the UAVs can be improved significantly.

We use the above four algorithms (bilateral filter, Otsu, YOLO, and FFT) to process two UAV tasks to verify the performance of the CPU/FPGA/NPU heterogeneous system: 1) process images of size 650×433 iteratively 1000 times and 2) process the data with 1 GB via the FFT. The experiments are conducted on three different heterogeneous computing platforms: 1) CPU/FPGA; 2) CPU/NPU; and 3) CPU/FPGA/NPU. The results show that the total times of running these tasks on CPU/FPGA and CPU/NPU for these two tasks are 29783 ms and 38267 ms, respectively. However, the time decreases to 23896 ms when these algorithms are executed on the CPU/FPGA/NPU heterogeneous system. This verifies the effectiveness of using heterogeneous systems with multiple kinds of accelerators to enhance UAV computing performance.

IX. CONCLUSION

This article presents the work of using a heterogeneous computing architecture to optimize both the completion time and energy cost of computationally intensive tasks on resource-constrained UAVs. A set of representative computationally intensive UAV tasks are deployed on heterogeneous systems, such as CPU/GPU, CPU/FPGA, and CPU/NPU, and the completion time and energy cost of running these tasks on these heterogeneous systems are measured and compared. From the results of this article, we can draw the following conclusions.

1) Different processors have different computing architectures and are, therefore, suitable for performing different types of tasks. Building a heterogeneous system with multiple kinds of accelerators can play a significant role in optimizing both the completion time and energy cost of computationally intensive tasks.
2) A computing system that can achieve the optimal task completion time may not have the lowest energy cost. Therefore, the management of a heterogeneous system needs to consider the tradeoff between completion time and energy consumption.
3) GPUs and NPUs both have high-parallel computing capabilities. However, algorithms that mostly perform sequential calculations, e.g., sorting in sequence, cannot fully utilize the computing advantages of GPUs and NPUs and, therefore, cannot demonstrate good computing performance on these cores.
4) The NPU has powerful 3-D cube computing capabilities. However, this advantage cannot be fully exploited when the computational complexity of the tasks is not high. Specifically, the NPU is more suitable for performing high-intensity computing tasks than computationally sparse tasks.
5) A method that shows advantages in one aspect may have disadvantages in another aspect, e.g., the NPU can increase the computing speed through low-precision FP16 operations, yet this can cause the problem of numerical overflow.
6) The hardware architectures of the GPU and NPU are fixed and designed specifically for certain types of

tasks. This feature makes them unsuitable for special operations, such as sequential sorting and large-scale accumulation. However, an FPGA is hardware reconfigurable and can design accelerators specifically for a given operation, which makes an FPGA suitable for use as a supplement to GPUs and NPUs in heterogeneous computing systems.

REFERENCES

[1] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, "Towards the unmanned aerial vehicles (UAVs): A comprehensive review," *Drones*, vol. 6, no. 6, p. 147, 2022.

[2] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Commun. ACM*, vol. 62, no. 2, pp. 48–60, 2019.

[3] B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, and V. J. Reddi, "Accelerating robot dynamics gradients on a cpu, GPU, and FPGA," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2335–2342, Apr. 2021.

[4] H. Liao, J. Tu, J. Xia, and X. Zhou, "DaVinci: A scalable architecture for neural network computing," in *Proc. Hot Chips Symp.*, 2019, pp. 1–44.

[5] Y. Tang and C.-l. Wang, "Performance modeling on DaVinci AI core," *J. Parallel Distrib. Comput.*, vol. 175, pp. 134–149, May 2023.

[6] T. Chen et al., "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Comput. Architect. News*, vol. 42, no. 1, pp. 269–284, 2014.

[7] D. Liu et al., "Pudiannao: A polyvalent machine learning accelerator," *ACM SIGARCH Comput. Architect. News*, vol. 43, no. 1, pp. 369–381, 2015.

[8] T. Luo et al., "DaDianNao: A neural network supercomputer," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 73–88, Jan. 2017.

[9] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, "DianNao family: Energy-efficient hardware accelerators for machine learning," *Commun. ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[10] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architect.*, 2017, pp. 1–12.

[11] A. S. Chakravarthy, S. Sinha, P. Narang, M. Mandal, V. Chamola, and F. R. Yu, "DroneSegNet: Robust aerial semantic segmentation for UAV-based IoT applications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4277–4286, Apr. 2022.

[12] D. Jaiswal and P. Kumar, "Real-time implementation of moving object detection in UAV videos using GPUs," *J. Real-Time Image Process.*, vol. 17, pp. 1301–1317, Oct. 2020.

[13] J. Wang, Y. Li, R. Li, H. Chen, and K. Chu, "Trajectory planning for UAV navigation in dynamic environments with matrix alignment Dijkstra," *Soft Comput.*, vol. 26, no. 22, pp. 12599–12610, 2022.

[14] M. H. Mousa and M. K. Hussein, "Efficient UAV-based MEC using GPU-based PSO and Voronoi diagrams," *CMES-Comput. Model. Eng. Sci.*, vol. 133, no. 2, pp. 413–434, 2022.

[15] M. Huang, H. Li, Y. Zhou, T. Ma, J. Su, and H. Zhou, "A UAV aided lightweight target information collection and detection approach," *Peer-to-Peer Netw. Appl.*, 2024, to be published.

[16] Y. Tao, R. Ma, M.-L. Shyu, and S.-C. Chen, "Challenges in energy-efficient deep neural network training with FPGA," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 400–401.

[17] H.-s. Suh, J. Meng, T. Nguyen, V. Kumar, Y. Cao, and J.-S. Seo, "Algorithm-hardware co-optimization for energy-efficient drone detection on resource-constrained FPGA," *ACM Trans. Reconfig. Technol. Syst.*, vol. 16, no. 2, pp. 1–25, 2023.

[18] Z. Zhang, M. P. Mahmud, and A. Z. Kouzani, "FitNN: A low-resource FPGA-based CNN accelerator for drones," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21357–21369, Nov. 2022.

[19] V. Sadhu, K. Anjum, and D. Pompili, "On-board deep-learning-based unmanned aerial vehicle fault cause detection and classification via FPGAs," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 3319–3331, Aug. 2023.

[20] Q. He, W. Chen, D. Zou, and Z. Chai, "A novel framework for UAV returning based on FPGA," *J. Supercomput.*, vol. 77, pp. 4294–4316, May 2021.

[21] A. Amanatiadis, L. Bampis, E. G. Karakasis, A. Gasteratos, and G. Sirakoulis, "Real-time surveillance detection system for medium-altitude long-endurance unmanned aerial vehicles," *Concurr. Comput., Pract. Exp.*, vol. 30, no. 7, 2018, Art. no. e4145.

[22] X. Li, Y. Wei, J. Li, W. Duan, X. Zhang, and Y. Huang, "Improved YOLOv7 algorithm for small object detection in unmanned aerial vehicle image scenarios," *Appl. Sci.*, vol. 14, no. 4, p. 1664, 2024.

[23] T. Tan and G. Cao, "Deep learning video analytics through edge computing and neural processing units on mobile devices," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1433–1448, Mar. 2023.

[24] X. Wang, S. Lai, Z. Chai, X. Zhang, and X. Qian, "SPGNet: Serial and parallel group network," *IEEE Trans. Multimedia*, vol. 24, pp. 2804–2814, Jun. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9454285

[25] S. Kim, B. Jang, J. Lee, H. Bae, H. Jang, and I.-C. Park, "A CNN inference accelerator on FPGA with compression and layer-chaining techniques for style transfer applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 4, pp. 1591–1604, Apr. 2023.

[26] J. Fowers et al., "Inside project Brainwave's cloud-scale, real-time AI processor," *IEEE Micro*, vol. 39, no. 3, pp. 20–28, Jun. 2019.

[27] Z. Que et al., "Remarn: A reconfigurable multi-threaded multi-core accelerator for recurrent neural networks," *ACM Trans. Reconfig. Technol. Syst.*, vol. 16, no. 1, pp. 1–26, 2022.

[28] J. Kim and S. Ha, "Energy-aware scenario-based mapping of deep learning applications onto heterogeneous processors under real-time constraints," *IEEE Trans. Comput.*, vol. 72, no. 6, pp. 1666–1680, Jun. 2023.

[29] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, "Deep learning inference parallelization on heterogeneous processors with TensorRT," *IEEE Embed. Syst. Lett.*, vol. 14, no. 1, pp. 15–18, Mar. 2022.

[30] O. Spantidi et al., "Targeting DNN inference via efficient utilization of heterogeneous precision DNN accelerators," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 1, pp. 112–125, Mar. 2023.

[31] T. Yang et al., "KubeEdge wireless for integrated communication and computing services everywhere," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 140–145, Apr. 2022.

[32] X. Liu, Q. Wang, C. Zou, M. Yu, and D. Liao, "Edge intelligence for smart airport runway: Architectures and enabling technologies," *Comput. Commun.*, vol. 195, pp. 323–333, Nov. 2022.

[33] "Ai tookit over KubeEdge." Sedna. Feb. 2024. [Online]. Available: https://github.com/kubeedge/sedna.

[34] A. D. Boursianis et al., "Internet of Things (IoT) and agricultural unmanned aerial vehicles (UAVs) in smart farming: A comprehensive review," *Internet Things*, vol. 18, May 2022, Art. no. 100187.

[35] H. Yao, R. Qin, and X. Chen, "Unmanned aerial vehicle for remote sensing applications—A review," *Remote Sens.*, vol. 11, no. 12, p. 1443, 2019.

[36] X. Liu et al., "Collaborative edge computing with FPGA-based CNN accelerators for energy-efficient and time-aware face tracking system," *IEEE Trans. Comput. Soc. Syst.*, vol. 9, no. 1, pp. 252–266, Feb. 2022.

[37] X. Liu, M. Zhang, C. Zou, J. Yang, and X. Yan, "Edge intelligence for smart metro systems: Architecture and enabling technologies," *IEEE Netw.*, vol. 36, no. 1, pp. 136–143, Feb. 2022.

[38] X. Liang, *Ascend AI Processor Architecture and Programming: Principles and Applications of CANN*. Amsterdam, The Netherlands: Elsevier, 2020.

[39] L. Chen, *Deep Learning and Practice With Mindspore*. Singapore: Springer, 2021.

[40] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, *Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems Using C++ and SYCL*. Berkeley, CA, USA: Springer, 2021.

[41] W. Feng, R. Maghareh, and K.-T. A. Wang, "Extending DPC++ with support for Huawei ascend AI Chipset," in *Proc. Int. Workshop OpenCL*, 2021, pp. 1–4.

Xing Liu received the Ph.D. degree from the School of Electronic Information, Wuhan University, Wuhan, China, in 2014.

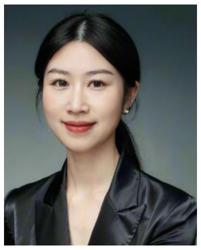
He is currently a Professor with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan. His research interests include the Internet of Things, artificial intelligence, and embedded systems.

Wenxing Xu received the bachelor's degree from the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China, in 2022, where he is currently pursuing the master's degree.

His research interests include Internet of Things and embedded systems.

Qing Wang received the bachelor's and master's degrees from the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China, in 2021 and 2024, respectively.

His research interests include Internet of Things, artificial intelligence, and embedded systems.



Mengya Zhang received the master's degree from Wuhan University of Technology, Wuhan, China, in 2013.

She is currently an Associate Professor with the School of Transportation and Logistics Engineering, Wuhan University of Technology. Her research interests include the intelligent transportation, Internet of Things, and embedded systems.