Hardware and Software Co-design for Soft Switch in ViT Variants Processing Unit

Wei Hu[1,2], Jie Fan[1,2](✉), Fang Liu[3,4], and Kejie Hu[1,2]

[1] College of Computer Science, Wuhan University of Science and Technology, Wuhan, China
{huwei,hekejie}@wust.edu.cn, JacobEvans.7@outlook.com

[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, China

[3] School of Computer Science, Wuhan University, Wuhan, China
liufangfang@whu.edu.cn

[4] Department of Information Engineering, Wuhan Institute of City, Wuhan, China

Abstract. As the application of pure Transformer in CV field, ViT shows the generality of Transformer model. However, it requires costly training on large datasets. Recently, some researchers trying to improve the training efficiency of ViT by combining ViT and CNN together which will use the inductive bias of CNN. In these models, the MHSA layer carries other modules on its side, but existing architectures cannot take advantage of this feature to customize designs to improve computational efficiency and resource utilization. The use of FPGA to customize specialized computing units can meet this need, but the existing hardware computing units can't adapt to the combination of different types of layers, and switching between different models will result in expensive re-production costs. In this paper, we use hardware and software cooperation to design the FPGA computing unit and divide the layers according to their functions. Convolution and Transformer are classified into one category. Under the coordination deployment of software, it mix the outputs of the same type of layers through soft switches, so as to adapt to those flexible models. Compared with the performance of the original model on CPU, it achieves the acceleration performance of 26× under the condition that the accuracy is only decreased by 0.9%. And the structure of common data block reduces the size of hardware resource unit by 91.7%.

Keywords: FPGA · CNN · Transformer · Deep learning · Hardware

1 Introduction

Since Transformer [1] was proposed, its variants shine in many fields, among which ViT [2] has achieved remarkable results in CV field. However, with the increase of training volume caused by the characteristics of Transformer itself, there are high requirements on the scale of model base. Excessive training volume also greatly increases the cost. Some researchers try to take advantage of CNN's high training efficiency and combine it with Transformer to meet the requirements of both accuracy and training efficiency. Influenced by this idea, ConViT [3], DeiT [4], CeiT [5] and LocalVit [6] models were gradually developed. They have different mixing modes, but when GPU is used to accelerate calculation, due to its universal characteristics, the model is not optimized, and the parallelism of the model is obviously unable to be utilized. Meanwhile, the volume and power consumption have always been difficult to be solved by this kind of universal computing unit. Customized FPGA can solve the above problems, and more and more researchers are trying to customize the calculation module or deploy the whole calculation model on FPGA [8,9].

There are now many variants of ViT, and it's very expensive to design dedicated computing units for each single variant, and rapid iterations of models do not wait for delays in hardware design. In order to bring production environments online quickly, there is a use of Vitis-AI technology to help quickly load custom models on hardware, but this solution has limited optimization. The inability to optimize data transfer between neurons and the need for specific DPU hardware accelerators greatly limit the minimum size of FPGA cells, losing one of the original intentions of using FPGA to customize the original.

In this paper, based on the characteristics of the current variants of Vit, and using the ConViT [3] model as a typical example, we designed a framework to simultaneously compute CNN and ViT (which are two neurons involved in most mainstream ViT variants) in one cell and output using soft switch control. In addition to speeding up the hardware through data quantization, neuron optimization, reordering of computing units and other steps, according to the analysis report, combined units are designed with software and hardware co-optimization [10–12] under the mechanism of soft switch and data sharing [13–15], which achieve better performance in computing efficiency [16–18] and resource utilization than single units.

1.1 Contribution

The main contributions of this paper are as follows:

- The CNN and ViT units are calculated simultaneously on FPGA device with each module being analyzed and optimized.
- Through the method of hardware and software co-design, it improves the utilization rate of Convolution module and Transformer module, and optimizes the generality of their combined units. Which makes models using CNN and VIT with minimal hardware resources to achieve great performance improvement.
- It is verified on the FPGA development board, and compared with the split implementation of CPU, GPU and FPGA model, the performance of our design is improved by 26×, 1.6× and 3.3×.

1.2 Related Work

In the application of Transformer in CV field, there are many work combining ViT and CNN. ConViT [3] introduced a new self-attention layer, GPSA *(Gated Positional Self-Attention)*, replacing some OF the SA layers with GPSA, based on ViT. ConViT can be initialized like CNN. Each attention head has a gating parameter to adjust expressiveness, which can be adjusted between local and global features. This parameter allows the model to decide for itself whether to maintain convolution.

$$A_{ij}^{h} := (1 - \rho(\lambda_h))softmax(Q_i^h K_j^{hT}) + \rho(\lambda_h)softmax(v_{pos}^{hT} r_{ij}) \tag{1}$$

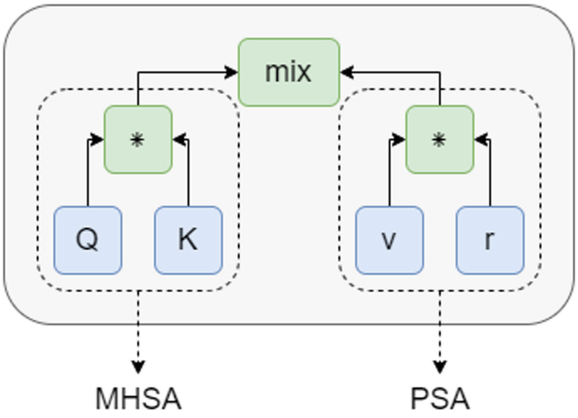

Fig. 1. Internal structure of GPSA

The author calls the part of simulated CNN PSA, as shown in Fig. 1, with the intention of introducing the locality of CNN into ViT, while the other part is the original MHSA. This study proves that ViT variants can achieve better results by absorbing the advantages of other modules on the basis of keeping the original module unchanged.

2 Background

2.1 ViT

The starting point of ViT is to completely replace CNN with attention. Although transformer was introduced in the CV field before, CNN or RNN was more or less used. ViT directly uses pure Transformer structure and has achieved good results. In the overall implementation, ViT completely uses the original Bert transformer structure, mainly to convert images into token like processing, and introduces the concept of patch, that is, the input images are divided into one patch after another, and then for each patch conversion (mainly flatten operation), Convert to a Bert-like input structure. The emergence of ViT marks the generality of transformer model and points the way for the unified universal encoder.

2.2 Relative Position Coding

However, at the beginning of the ViT, the SA layer does not know the local relationships of patches, but according to the CNN model, it is obvious that this is relevant, which also leads to low training efficiency of the ViT model. The introduction of relative position coding solves this problem well. Different from general SA, PSA(positional self-attention) [7] is added to encode relevant information at different positions.

$$A_{ij}^{h} := softmax(Q_i^h K_j^{hT} + v_{pos}^{hT} r_{ij})$$

Each self-attention head uses a trainable embedded position code v and relative position code information r, and only relies on the distance between pixels to simulate the effect of CNN.

3 Design Scheme Exploration

In this section, we will introduce the optimization steps of Convolution and Transformer modules deployed on FPGA under the co-design of hardware and software. Firstly, the parameter quantization scheme is determined based on the commonality of the two modules, and then model by model analysis.

3.1 Parameter Quantitative

In the deep learning model, reducing a certain precision will only bring a very small loss of the overall precision of the model [19]. The performance of FPGA is very sensitive to the data transmission rate, that is, a certain loss of accuracy can be used to exchange for a lower delay of FPGA as a whole. Therefore, reasonable quantization is necessary to achieve a balance of performance, precision and hardware utilization [20–22].

Some studies [23,24] show that when the accuracy is reduced to 8 bits, the accuracy of ViT model and CNN model is reduced by less than 1%. We used data quantization of different precision and computational quantization of 8 and 32 bit integer types for quantitative analysis of Convit. We ran the Convit-Tiny model to classify ImageNet data set, and the results were shown in Fig. 2.

The asymmetric quantization algorithm is used to carry out static quantization after the model training, and all tensors in the calculation process are adjusted according to the corresponding scale and zero point according to the input matching the classification test.

$$Q = round(\frac{input - zero_point}{scale})$$

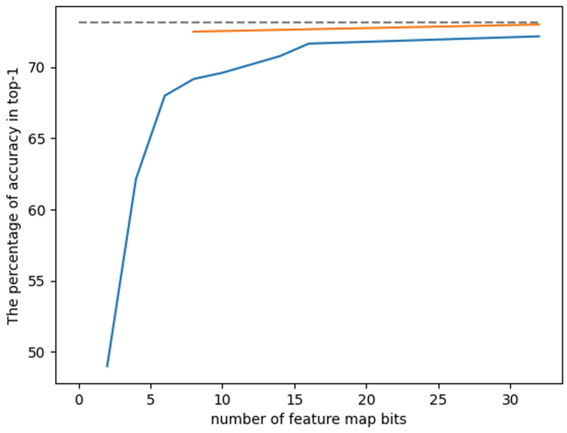

Fig. 2. As the bit width changes, the accuracy on ImageNet changes, the gray dotted line is the original accuracy, the orange is the calculated quantization, and the blue is the data quantization (Color figure online)

According to the quantization results, Q = 8 is the optimal quantization scheme that comprehensively considers the accuracy and resource utilization in scenarios that do not require extremely high accuracy. At the same time, when Q > 8, the classification accuracy is slightly improved while the performance is reduced. Therefore, if no other model is introduced, all experiments after this section will use 8-bit quantization scheme for data representation.

3.2 Model Analysis

An important theory of hardware acceleration is to accelerate high probability events, and the benefits of accelerating high probability events are far greater than those of other events. Therefore, we need to analyze each part of the model, find the commonness between modules, do common optimization, and optimize the different parts separately and calculate in parallel. To this end, we calculate the main Convit-Tiny calculations on FPGA and look for optimization breakthroughs from them.

As shown in Fig. 3, a large amount of time is spent on the convolution and Transformer modules of the model, and the time spent optimizing these two modules will play a decisive role in our overall performance. By default, pipeline optimization has been carried out for these two modules in our subsequent experiments. At the same time, we can also find that the remaining two modules consume very short time, so it is not cost-effective to optimize such a neglected module. However, we can see from the parameter information that their input size is similar to that of the GPSA module. If we ignore them here, we will lose a large amount of data space, which is also unacceptable. To do this, we need to find ways to make temporally negligible modules possible in space. This leads to the key of this article, the soft switch.

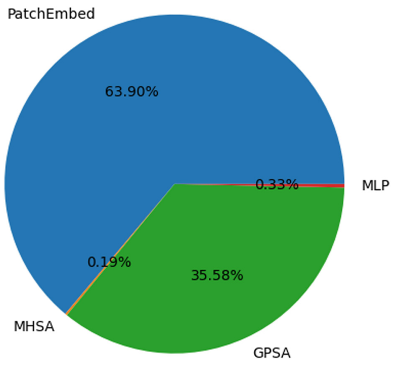

Fig. 3. Cumulative time ratio of each module when Convit-Tiny runs ImageNet data set

3.3 Soft Switch and Data Sharing

Previously, researchers used hard switches on hardware to switch between modules, which we called hard switches, that is, hard coded hardware before deployment. The hardware implemented by the hard switch is immutable in the execution order of each module at runtime, which brings the benefit of maximizing the performance by optimizing unit layout and rearranging pipeline during the coding phase. However, this also brings the defect of almost non-dynamic transformation, which is also the defect of FPGA itself, each architecture adjustment needs to regenerate the corresponding firmware.

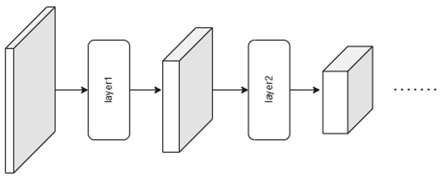

Fig. 4. Data transmission mode under hard switch

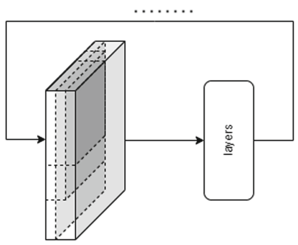

Fig. 5. Data transmission mode under soft switch

Hard switches are implemented knowing the order of modules in advance, and in order to achieve reusable designs, the conventional approach is that modules are independent of each other, and the optimization of modules only exists internally. The final data transfer flow and logic are consistent, as shown in Fig. 4. Now we need each module to reuse data blocks, so we do not want each module to be independent of each other, and this hard switch design idea hinders this, so we need to design a soft switch architecture.

As shown in Fig. 5, in the soft switch mode, all the input data in a data pool, at the beginning of the design and analysis of the maximum data needed for the scale (generally speaking is the original length $\times$ original width $\times$ max number of channel), run time, to the size of the data pool filled with dynamic data block, and specialized in counting read this part is used to calculate. The Layers part is almost identical to the hard switch mode, but transforms the corresponding computing module based on the input. This dynamic way of filling, reading, and calculating data is almost impossible to implement in hardware, which is why previous designs used hard switches, but software and hardware work together to achieve this. The next section describes how to use hardware and software collaboration to cover all computing modules in the same cell and how to share data blocks.

4 Hardware and Software Co-design

Based on the analysis in the previous section, we need not only to transmit quantization data to the hardware computing unit, but also to tell the hardware the mixing ratio of the two models. Software and hardware are not independent. Therefore, in addition to their own internal optimization, both software and hardware need to conduct collaborative design optimization for their interaction. In terms of software, in order to cooperate with hardware to do general processing, data flow needs to do general processing, that is, data flow has nothing to do with the shape and size of data block; For different mixing ratios, when one side approaches 0, the jump pulse is released, and the asynchronous data receiving wait for the model is closed when the data is received.

In terms of hardware, it is analyzed that CNN and Transformer need common data block and create common data pool. The convolution operation is optimized by loop unrolling and flow rearrangement. The data transformation analysis of self-attention operation is carried out, and the relative position coding matrix which changes with the size of matrix is loaded as the weight, the fixed position coding information is hard coded, and the matrix multiplication is divided into data blocks to optimize the occupancy of hardware resources.

4.1 Software Layer Design

In the transmission part, in order to be compatible with matrices of different sizes, one-dimensional mapping of multidimensional matrices is required, and then reconstructed by hardware.

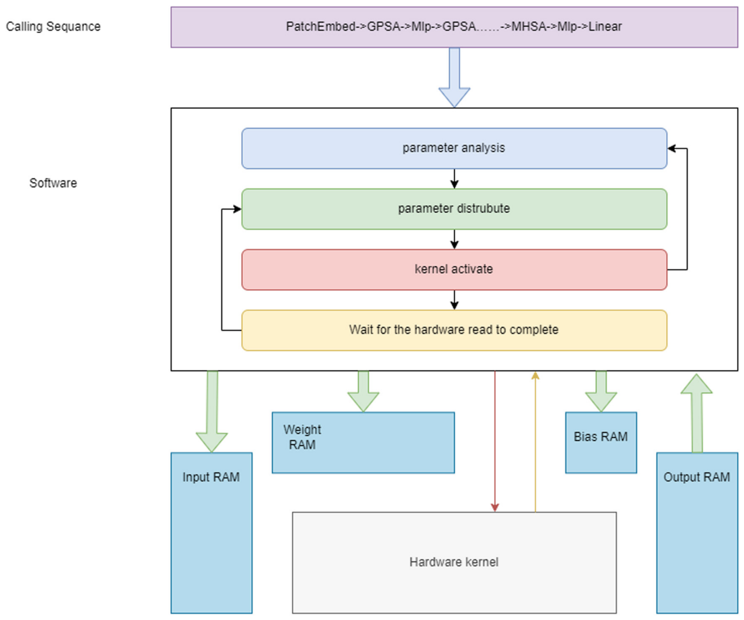

Fig. 6. Data flow between software layer and external and internal business logic

In order to enable the hardware unit of the driver to handle different types and sizes of input, a separate software layer is used to dynamically distribute data and send corresponding activation signals to the hardware. The software layer is divided into four stages: parameter preprocessing, parameter distribution, activation of the hardware core, waiting for the core to finish reading data. After the core is read, it will enter the next distribution. In general hardware accelerators, the software layer only acts as data transmission, which is equivalent to repeatedly doing the parameter distribution stage in this paper, that is, continuously feeding weights to the hardware end without changing the execution order of the model according to the specific situation, but the implementation of this paper will be segmented to each Block.

As shown in Fig. 6, when a call is initiated,

- the software layer enters the parameter analysis stage, from which it reads the corresponding module information, obtains the parameters, and converts them into the corresponding software parameter configuration to prepare for data transmission.
- Then the data is transmitted, the multi-dimensional matrix is flattened into one dimension and continuously sent to the hardware core through the flow channel, and the hardware model type is informed, and the corresponding parameter recombination unit and calculation unit are activated.
- The latter stages of the parameter distribution and active hardware for parallel execution, activate the signal immediately after resolving the parameters

of the next Block, and wait for the last round of parameters used by all finished, hardware return after reading complete signals, said the data in RAM can be covered.

A layer of software operation on end, began to continue loading parameters of the round.

4.2 Hardware Layer Design

In general, each cell manages its own block of data, including loading (external input), calculation, and output. This allows all cells to be block-sized in advance on demand, but it also creates a problem because the algorithm flow is fixed and any software changes require tweaking the entire hardware core. We use the data pool inside the hardware unit to store all the data, whose size is related to the maximum amount of data required for a single calculation. The software layer informs the calculation scale, and the data is hashed during actual storage.

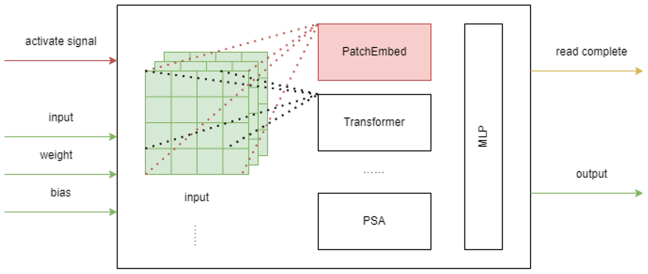

Fig. 7. shows the data flow direction and data block mapping when PatchEmbed's module is activated

As shown in Fig. 7, when an activation signal from the software layer is input into the hardware, the corresponding module is activated, and the effective data area of the data pool is divided according to the preset parameter size of the module, and input, weight and bias are read. Taking input as an example, the convit-Tiny model firstly divides the 3-channel image 224×224 into 196 patches of 192 size by patchEmbed module. At this time, the input of patchEmbed module is $3 \times 224 \times 224$, that is, the size of data block is at least $3 \times 224 \times 224$. Observe that the size of subsequent input and output data blocks does not exceed this size (for example, the input of *gate position self-attention* (GPSA) module is 196×192, and the input of multi-head self-attention (MHSA) module is 197×192).

The data usage is shown in Fig. 8. We can use a matrix with a maximum of $3 \times 224 \times 224$ to hold all matrices smaller than it, and only receive the required dimensions when receiving data. Extra space is skipped as padding, and extra dimensions and padding are ignored in the calculation.

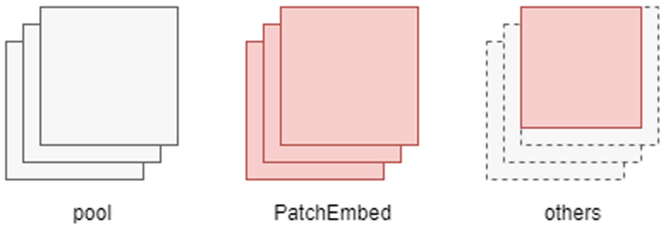

Fig. 8. Usage of data pools by different computing modules

5 Experiments

5.1 Experimental Setup

This paper uses Vivado *Hight-Level Synthesis* (HLS) and uses commonly used ImageNet data sets for validation. The hardware and software co-design unit is equipped with Xilinx ZCU102 development board, and has an ARM core running software layer and a FPGA programmable array running hardware layer. The CPU and GPU models were built using PyTorch. The CPU model is 11th Gen Intel(R) Core(TM) I5-1135G7 @ 2.40 ghz 2.42 ghz, and the GPU model is NVIDIA GeForce RTX3060. All reasoning times are the average values of 50000 images in all validation sets.

5.2 Experimental Results

In order to show the improvement of optimization in this paper intuitively, we will carry out combinational optimization on the original model successively, including the flow rearrangement, parameter quantization and software/hardware coordination mentioned in the previous chapter.

As can be seen from Fig. 9, quantization brings the largest increase in the size of input data, while pipeline rearrangement brings the largest performance improvement. The combination of pipeline rearrangement and quantization alone can bring 70.3% performance acceleration and 87.1% reduction in the size of input data. Full-text soft switch is the key to design at the same time, it can give the cell dynamic ability is derived, of course, the design of hardware and software collaborative also partly destroyed the oneness of hardware, software is the concurrent design reduces the affect of this operation on performance, makes the performance loss caused by the negligible, and to a certain extent reduce the data size, about a third.

Overall performance and data occupancy, as well as dynamics, this paper finally adopted a combination of all three optimizations, achieving a 69.5% performance improvement and a 91.4% reduction in input data size. Based on the above optimization combinations, we selected all optimization combinations as the final results of this paper and compared them with running the same parameters on CPU and GPU using the original PyTorch version.

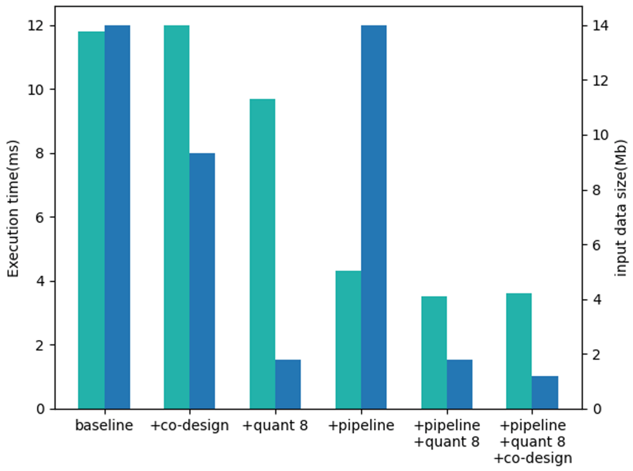

Fig. 9. Performance and data size under different combinatorial optimizations

Table 1. Comparison with CPU and GPU

Device	CPU	GPU	FPGA
Execution time (ms)	93.7	4.5	3.6
Parameters (M)	5.7	5.7	5.7
Acc@1	73.1	73.1	72.4

As can be seen from Table 1, FPGA can still achieve better performance than GPU under the condition of limited hardware resources and dynamic capability of hardware, while the accuracy is only reduced by 0.9%, proving that the soft switch architecture proposed in this paper is sufficient to cope with the variable model.

6 Conclusion

Based on the model combined with CNN and Transformer, this paper took Convit as an example and tried to solve the inflexible defects of current hardware development by means of soft switching and data sharing on the basis of conventional quantization and rearrangement flow and designed a hardware and software co-architecture. We tested our ideas on the Xilinx ZCU102 development board, achieved performance exceeding GPU under the same model with extremely limited resources and using less than 10% of the data size compared to the original hardware.

References

1. Vaswani, A., Shazeer, N., et al.: Attention is all you need. In: Proceedings of the NIPS, vol. 30, pp. 5998–6008 (2017)
2. Dosovitskiy, A., Beyer, L., et al.: An image is worth 16×16 words: transformers for image recognition at scale. In: Proceedings of the ICLR (2021)
3. d'Ascoli, S., Touvron, H., et al.: ConViT: improving vision transformers with soft convolutional inductive biases. In: Proceedings of the ICLR, pp. 2286–2296 (2021)
4. Touvron, H., Cord, M., et al.: Training data-efficient image transformers and distillation through attention. In: Proceedings of the ICLR, pp. 10 347–10 357 (2021)
5. Yuan, K., Guo, S., et al.: Incorporating convolution designs into visual transformers. In: Proceedings of the ICCV, pp. 579–588 (2021)
6. Li, Y., Zhang, K., Cao, J., et al.: LocalViT: Bringing locality to vision transformers, arXiv:2104.05707 (2021)
7. Ramachandran, P., Parmar, N., Vaswani, A., et al.: Stand-alone self-attention in vision models, arXiv preprint arXiv:1906.05909 (2019)
8. Yang, Y., Huang, Q., et al.: Synetgy: Algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs, November 2018
9. Hu, W., Chen, S., Li, Z., Liu, T., Li, Y.: Data optimization CNN accelerator design on FPGA. In: IEEE ISPA/BDCloud/SocialCom/SustainCom, pp. 294–299 (2019)
10. Qiu, M., Li, H., Sha, E.: Heterogeneous real-time embedded software optimization considering hardware platform. In: ACM Symposium on Applied Computing, pp. 1637–1641 (2009)
11. Qiu, M., Xue, C., et al.: Energy minimization with soft real-time and DVS for uniprocessor and multiprocessor embedded systems. In: IEEE DATE, pp. 1–6 (2007)
12. Liu, M., Zhang, S., et al.: H infinite state estimation for discrete-time chaotic systems based on a unified model. IEEE Trans. SMC (B) **44**, 155–168 (2012)
13. Wu, G., Zhang, H., et al.: A decentralized approach for mining event correlations in distributed system monitoring. JPDC **73**(3), 330–340 (2013)
14. Qiu, M., et al.: Data transfer minimization for financial derivative pricing using Monte Carlo simulation with GPU in 5G. J. Commun. Syst. **29**(16), 2364–2374 (2016)
15. Qiu, L., Gai, K., Qiu, M.: Optimal big data sharing approach for tele-health in cloud computing. In: IEEE SmartCloud, pp. 184–189 (2016)
16. Qiu, M., Xue, C., et al.: Efficient algorithm of energy minimization for heterogeneous wireless sensor network. In: IEEE EUC, pp. 25–34 (2006)
17. Qiu, M., Guo, M., et al.: Loop scheduling and bank type assignment for heterogeneous multi-bank memory. JPDC **69**(6), 546–558 (2009)
18. Li, J., Qiu, M., et al.: Thermal-aware task scheduling in 3D chip multiprocessor with real-time constrained workloads. ACM TECS **12**(2), 1–22 (2013)
19. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave Gaussian quantization. In: IEEE CVPR (2017)
20. Qiu, M., Chen, Z., et al.: Data allocation for hybrid memory with genetic algorithm. IEEE TETC **3**(4), 544–555 (2015)
21. Qiu, M., Gai, K.: Heterogeneous assignment of functional units with Gaussian execution time on a tree. In: IEEE 20th Conference on HPCC (2018)
22. Qiu, M., et al.: Cost minimization for heterogeneous systems with Gaussian distribution execution time. In: IEEE 17th Conference on HPCC (2015)

23. Liu, Z., Wang, Y., Han, K., Ma, S., Gao, W.: Post-Training Quantization for Vision Transformer, arXiv:2106.14156 (2021)
24. Gysel, P., Pimentel, J., Motamedi, M., Ghiasi, S.: Ristretto: a framework for empirical study of resource-efficient inference in convolutional neural networks. IEEE Trans. Neural Netw. Learn. Syst. **29**, 5784–5789 (2018)