Received April 12, 2019, accepted May 3, 2019, date of publication May 7, 2019, date of current version May 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2915380

GPU Energy Consumption Optimization With A Global-Based Neural Network Method

YANHUI HUANG[1], (Member, IEEE), BING GUO[1], (Fellow, IEEE), AND YAN SHEN[2], (Fellow, IEEE)
[1]College of Computer Science, Sichuan University, Chengdu 610065, China
[2]School of Control Engineering, Chengdu University of Information Technology, Chengdu 610042, China

Corresponding author: Bing Guo (guobing@scu.edu.cn)

This work was supported in part by the State Key Program of National Natural Science Foundation of China under Grant No. 61332001; The National Natural Science Foundation of China under Grant No. 61772352, 61472050; the Science and Technology Planning Project of Sichuan Province under Grant No. 2018ZDZX0010, 2017GZDZX0003.

ABSTRACT With the widespread use of smart technologies, graphics processing unit (GPU) power-optimization issues are becoming increasingly important. Many researchers have tried to use dynamic voltage and frequency scaling (DVFS) technology to optimize a GPU's internal energy consumption. However, DVFS energy management often has difficulty balancing GPU performance and energy efficiency. This paper aims to implement a DVFS energy management strategy. We constructed a new type of neural network to a GPU-based energy management scheme, implemented the global-based DVFS model, and explored its implementation details. Using a master-slave model, we built a global energy control solution strategy. This strategy performs global collaborative DVFS adjustments on the GPU's energy consumption module based on task characteristics. Through the software construction and implementation of the global-based DVFS model, we proved that the strategy improves the GPU performance while improving the GPU's energy efficiency. We conducted performance and energy tests on three GPUs on the Tesla, Fermi, and Kepler platforms. The experiments showed that this strategy improved the performance and power consumption of GPUs based on each of the platforms.

INDEX TERMS Collaborative control, DVFS method, global-based energy optimization, task feature.

I. INTRODUCTION

High-performance computing is evolving in clustering and massive parallelization, especially involving machine learning, which increases the computational complexity and dramatically increases the power consumption of the entire graphics processing unit (GPU). Therefore, how to improve the energy consumption performance of a GPU so that it can adapt to this increasing power consumption is especially important. Dynamic voltage and frequency scaling (DVFS) energy management technology is one of the most promising GPU power-management strategies. DVFS is an efficient way to manage energy by dynamically adjusting the voltage and frequency parameters of the GPU for energy management. With specific tools, researchers can continuously adjust the core voltage and frequency in the GPU [1].

DVFS has proven to be an effective CPU power-saving method [2]. While DVFS technology is still in its early stages, compared to the mature CPU voltage- and frequency-adjustment technology, the DVFS strategy of GPU energy consumption is still relatively less mature. DVFS for GPUs is often based on simple strategies such as defining four P-states [3] on the GeForce GTX 980, each a specific combination of GPU operating voltage and frequency. According to the working environment parameters of the GPU, i.e. the task load in the GPU, the GPU is set to work in a certain state, and the energy consumption can be directly controlled. However, due to differences in GPU operating mechanisms, the energy management parameters in GPUs are insufficient compared to those within GPUs. Thus, it is not possible to simply transplant the DVFS energy management strategy in a CPU to a GPU. This creates a dilemma for DVFS management in GPUs, specifically, to increase productivity, you must increase power consumption, and vice versa. For example, in the P-state management policy just mentioned, the four P states are P0, P2, P5, and P8. P8 is an idle state in which energy consumption and work efficiency are low. P2 provides the highest voltage and frequency, and the GPU in this state is very efficient, but it consumes much energy. Therefore, how to improve the GPU's work efficiency while reducing its energy consumption is particularly meaningful.

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

Recently, researchers have conducted a lot of research based on DVFS GPU energy consumption. For example, Kim *et al.* [4] scaled the frequency of the GPU with various matrix multiplications on the NVIDIA Fermi platform to save energy. Abe *et al.* [5] combined the core and memory frequencies of GPUs based on Tesla, Fermi, and Kepler platforms to try to improve the performance and power consumption of the GPU. The experimental results showed that the frequency combinations had different energy consumption effects on different platforms. Currently, GPU DVFS energy optimization research faces the following challenges. First, a certain DVFS strategy may produce different results on GPUs on different platforms. Second, GPU hardware and power management information is very limited. Finally, experiments have lacked accurate quantitative power estimation.

The purpose of this paper is to analyze the causes of this phenomenon, and to use DVFS technology based on the new neural network to solve this dilemma, so as to achieve a management strategy that simultaneously improves GPU efficiency and reduces energy consumption. In this paper, we built a real DVFS measurement environment and obtained more accurate GPU performance data compared to software simulation method. At the same time, we compared the effects of GPU implementation energy optimization strategies on different platforms, and analyzed the effects of the energy optimization strategies proposed in this paper on different platforms.

In addition, current GPU energy management models often use neural networks to overcome the nonlinear relationship between GPU modules, but the traditional neural network is fully connected, which is complex. This is not conducive to real-time computing, and we need to establish better real-time control strategies for GPUs. This strategy is necessary to build an efficient GPU energy management system.

Through analysis, we have summarized two important strategies related to GPU energy management. The first is task-feature control; the task feature is highly correlated with the GPU's power consumption [6]. The task feature in the GPU can be used for energy consumption adaptive control. The second is based on a global collaborative energy management strategy. We use a proportional-integral-derivative neural network (PIDNN) with high real-time processing to realize the coordinated control of the GPU energy management modules with efficient real-time computing. We combine these two strategies to form a global-based PIDNN collaborative energy control strategy, i.e. the global-based PIDNN, and we use the CUDA programming environment to build a code framework. Hence, we increase GPU performance while reducing power consumption.

Our work incorporates the following two innovations.

1) A multi-variable, global-based DVFS energy management model is constructed. The model has the advantage of real-time control and has a task-tracking feature to realize the coordinated energy consumption control of each energy module in the GPU.

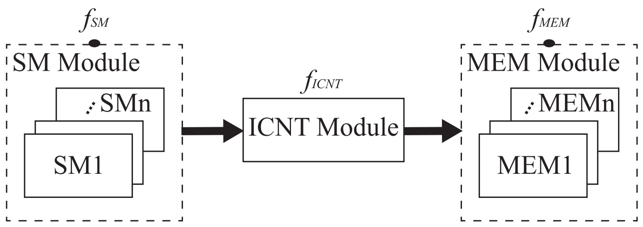


FIGURE 1. Architecture of GPU energy consumption modules.

2) We use the existing GPU software-implementation framework CUDA model to build a new energy optimization model called a master-slave-based model. We designed the model based on the idea of modularization and realized the code framework of the GPU energy management system through system-function division.

The remainder of this paper is organized as follows.

In the next part, we describe the overall energy consumption model in the GPU, analyze the dilemma faced by the DVFS energy management strategy, and propose to build a new energy management model with PIDNN. In the third part, we use the PIDNN model to construct the energy control module through the task feature. We use PIDNN to build a DVFS energy management solution for the GPU and, at the same time, build a master-slave model to implement a global-based energy optimization model. Then we train the model and elaborated on the algorithm process and the code implementation framework. In the fourth part, we carried out experimental simulations and verify the performance and energy optimization characteristics of the global-based DVFS model for three different GPU platforms.The fifth part of this paper summarizes our work and discusses possible future work.

II. BACKGROUND AND PRELIMINARY CONSIDERATIONS

A. GPU OVERALL ENERGY CONSUMPTION MODEL

The dynamic energy consumption of the GPU has multiple parts, as shown in Fig. 1. The GPU has three main energy consumption modules. These are the stream multiprocessor (SM), memory (MEM), and interconnection network (ICNT) modules. Each of these modules is explained below.

The SM module is composed of multiple stream processors (SMs) and L1 cache. The SM is the core component for calculation, and it includes several high-speed pipelines to complete high-speed calculations. From the perspective of energy consumption, because the module is responsible for high-speed computing functions, it consumes about 40% of GPU power [7].

The MEM module is a standalone global memory-management module composed of off-chip L2 cache. This module interacts with the SM calculation results, and probably uses more than 30% of GPU power.

The ICNT module is connected to the calculation and MEM modules; it is responsible for communication between the SM and MEM modules. It uses a cross-switch circuit to achieve high-speed communication bandwidth between modules. This module uses 20% of GPU power.

Based on the analysis of the global energy architecture, a GPU's total energy consumption consists of static energy consumption and dynamic energy consumption. The static energy consumption can be expressed by (1).

$$P_{static} = I_{static} \cdot V_{dd}. \tag{1}$$

In the above formula, P_{static} represents the static power consumption of the GPU, V_{dd} is the power supply voltage, and I_{static} is the total current flowing through the CMOS circuit. Static power is usually very low, so its impact is neglected in this article.

A GPU's dynamic energy consumption is the sum of the energy consumption of the three modules. The dynamic energy consumption of the GPU is represented by (2).

$$P_{dynamic} = \sum\nolimits_{i=1}^{3} a_i \cdot C_i \cdot V_i^2 \cdot f_i. \tag{2}$$

In the above formula, a_i is the active factor of each module; C_i is the capacitance parameter of each module; V_i is the voltage parameter of each energy consumption module; and f_i is the operating frequency of each module, whose three working frequencies are expressed as f_{SM}, f_{MEM}, and f_{ICNT}. Our goal is coordinated control over the operating frequencies of these three modules.

B. DVFS ENERGY MANAGEMENT STRATEGY DILEMMA

The DVFS energy management strategy is based on two energy consumption prediction models. One strategy is to build a power-consumption regression model for the GPU, which will be used to predict and manage GPU energy consumption. For example, a task feature model was built for the GPU, using DVFS control to predict and reduce the power consumption of the GPU [8]. An analysis based on kernel coarse-grained granularity and optimized GPU energy consumption was established [9]. The energy limitation method was used to perform DVFS control on the GPU to increase throughput [10].

We model GPU power consumption linearly as:

$$P = a_0 + a_1{}^{*}x_1 + a_2{}^{*}x_2 + \cdots + a_n{}^{*}x_n, \tag{3}$$

where P is the power consumption, $x_1, x_2, \ldots, x_n$ are the n input variables, and $a_0, a_1, \ldots, a_n$ are the output contributions. This strategy assumes that the energy consumption modules in the GPU are linear. The problem with this model is its inability to accurately predict nonlinear GPU energy systems.

Another method is to use an artificial neural network (ANN) to build a predictive model. Li *et al.* [11] used five system parameters to establish an energy consumption model for the GPU, and used DVSF technology to adjust its operating voltage. The advantage of modeling with the ANN method is its facility with the nonlinear relationship between variables. However, it requires a large amount of training data to build the model, and the output model has high complexity.

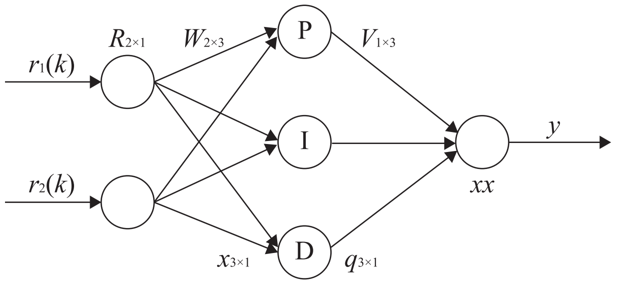


FIGURE 2. **PIDNN controller microarchitecture.**

The prediction error of the GPU regression model is often large. For energy consumption systems that require more efficient real-time features, the prediction accuracy of this modeling method is relatively low, and it is not possible to cooperatively control multiple energy consumption modules in a system. To achieve coordinated global management of multiple energy modules of the GPU requires a more effective prediction model.

C. PIDNN COLLABORATIVE CONTROL MODEL

Regarding the decoupling problem, Shu [12] and Liu and Song [13] proposed a proportional-integral-derivative neural network (PIDN) and demonstrated its cooperative control characteristics.

PIDNN is a special multivariable collaborative controller composed of a neural network and PID. Its network structure is shown in Fig. 2. Putting the PID neurons into the multi-layer forward network constitutes a univariate PIDNN controller that can serve as the basic control unit for the GPU. The basic form of the PIDNN controller is 2-3-1. It consists of two input-layer neurons, three hidden-layer neurons, and one output-layer neuron. The hidden-layer neurons are composed of proportional, differential, and integral elements.

The PIDNN controller works as follows. First, the output of the signal $r_1(k)$, $r_2(k)$ input-layer neurons enters the hidden layer through the first-level network W and the three neurons of the hidden layer. The element performs proportional, integral, and differential processing on the input signal. Then the input of the hidden layer enters the output layer through the second-level network V, and the neurons of the output layer complete the output y of the entire network with the proportional neurons. The PIDNN can complete the single-variable control task. The hidden-layer neuron functions of PIDNN are both static and dynamic. Their state functions are as follows:

$$\begin{cases} P: u_1(k) = x_1(k) \\ I: u_2(k) = q_2(k-1) + x_2(k) \\ D: u_3(k) = x_3(k) + x_3(k-1) \end{cases} \tag{4}$$

In the formula, the P neurons are static neurons, and the I and D neurons are dynamic neurons. The PIDNN controller incorporates the PID control law in the neural network. This causes the controlled parameters to only be related to the given control parameters, and has nothing to do with other

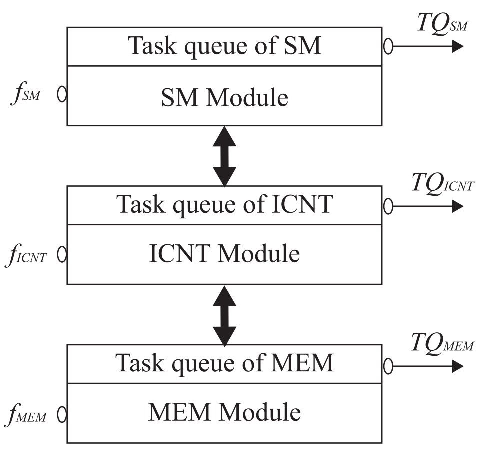


FIGURE 3. GPU global-based energy system architecture.

control parameters, so that the PIDNN has good cooperative control characteristics. In addition, PIDNN is not a fully connected network structure. Compared with ANN, the network parameter adjustment of this structure has better real-time performance [14].

III. METHODOLOGY

A. ENERGY MANAGEMENT MODEL BASED ON THE TASK FEATURE

The main energy-consumption modules of the GPU are the stream multiprocessor module (SM module), memory module (MEM module), and ICNT module. We abstract the energy module in the GPU and use Fig. 3 to illustrate it.

The operating frequencies of the three energy-consumption modules are independent. If we set the operating frequencies of the SM, MEM, and ICNT modules to be f_{SM}, f_{MEM}, and f_{ICNT}, respectively, the clock frequencies of the modules can be independently adjusted, and individual frequency control of the three modules is possible.

When the task access in the SM module fails, the task queue TQ_{CM} will be formed in the SM module. The length of the queue reflects the size of the SM workload. Similarly, when multiple SMs communicate with the MEM module, the ICNT module's task load will increase, and a task queue TQ_{ICNT} will be formed. The larger the load, the longer the task queue. By the same token, when the task of sending ICNT modules to the MEM module increases, the task queue of the MEM module TQ_{MEM} is also increased. When the GPU system has a memory-access request, if the cache-access fails, then the request queue is written to the miss status holding register (MSHR). The GPU can obtain the task queues of the three energy-consumption modules from the MSHR parameters, TQ_{SM}, TQ_{MEM}, and TQ_{ICNT}, and use them as parameters to control the operating frequency of the energy-consumption module.

Based on the above observations, we constructed a collaborative optimization method for the GPU energy-consumption module using the system-control characteristics of the PIDNN network. The system framework of this method is shown in Fig. 4.

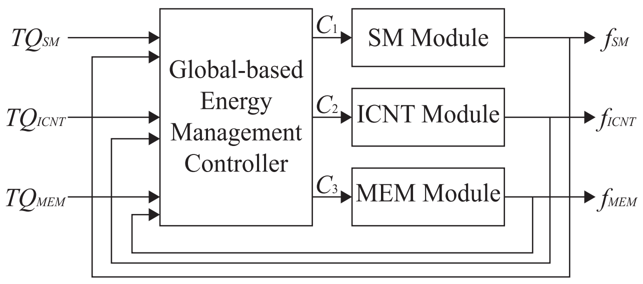


FIGURE 4. Global-based DVFS energy model.

At the heart of the global-based DVFS energy-management model is a global controller consisting of three sets of PIDNN networks. We input the task queue parameters TQ_{SM}, TQ_{MEM}, and TQ_{ICNT}, and the operating frequency parameters f_{SM}, f_{MEM}, and f_{ICNT} of the three energy-consumption modules to this global decoupling controller. The controller decouples the three sets of parameter signals TQ_{SM}, TQ_{MEM}, and TQ_{ICNT}, and outputs three working-frequency control signals C_1, C_2 and C_3 for real-time control of the operating frequency of the three modules, thereby realizing energy-consumption optimization. From Fig. 4, we can see that this control method comprehensively considers the control modules, and the energy-consumption optimization of the modules is independently related to each other. Since the method utilizes task-queue parameters TQ_{SM}, TQ_{MEM}, and TQ_{ICNT} to generate corresponding control signals, this adjustment has attributes based on task characteristics. We call this energy-management model the global-based DVFS energy-management
model.

B. GLOBAL-BASED DVFS MODEL TRAINING PROCESS

The global-based DVFS energy optimization model is based on the PIDNN controller, which consists of three sets of PIDNN controllers (Fig. 2). The core of each group of controllers is the P-I-D neuron function of the hidden layer. These three sets of controllers form a global-based energy optimization model through the V and W networks. Global-based DVFS adjusts the weight parameters of the V and W according to the gradient descent method, and gives the training results. The V and W weight parameters adjustment equation is the mean square error of the input signal:

$$J = \sum_{n=1}^{3} E_n = \frac{1}{l}\sum_{n=1}^{3}\sum_{k=1}^{l} \left[r_{n1}(k) - r_{n2}(k)\right]^2, \tag{5}$$

where J represents the mean square error of the input signal; r_{n1} and r_{n2} represent the input signals; l is the number of points used per batch; n represents the subnet number. Since the global-based DVFS model consists of three PIDNN subnets, $n = 1, 2, 3$.

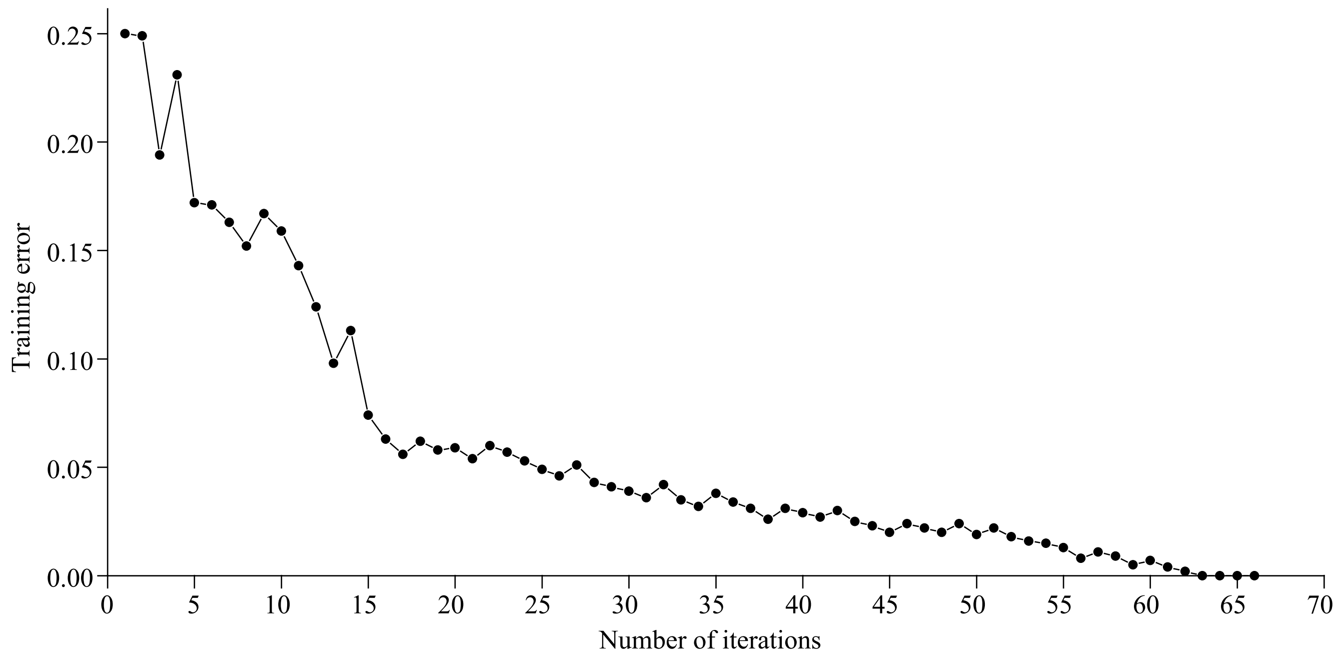


FIGURE 5. Global-based model training process.

$$W=\begin{bmatrix} 1.3643 & -0.7193 & -1.2834 & -1.2982 & 1.2937 & 1.8389 & 0.9695 & 0.8521 & 0.2512 \\ -0.8276 & 0.6293 & 0.2936 & -0.0817 & -1.8334 & 0.7163 & -1.4165 & 1.3231 & 0.7499 \\ 0.5281 & -0.2203 & -1.0287 & 0.6221 & 1.3944 & -1.6529 & 0.2064 & -1.5341 & -0.1485 \\ -0.3947 & 0.1287 & -0.8678 & 0.2695 & 0.8372 & 0.7512 & -1.9273 & -0.8237 & 0.1678 \\ -0.3875 & -0.3203 & -0.2387 & -1.2165 & 1.5475 & 0.4399 & 0.1883 & -1.1715 & -0.0287 \\ 0.2634 & 0.2827 & 0.4647 & 0.3064 & -0.7332 & 1.2654 & 0.9283 & 0.2275 & 0.5647 \end{bmatrix}, \quad V=\begin{bmatrix} 0.8173 & -0.9274 & 1.2823 \\ -1.3848 & 1.0273 & -1.5283 \\ 0.6539 & -1.3856 & 0.2736 \\ 0.4253 & 0.2506 & -0.7288 \\ -0.1678 & -0.7145 & 0.5225 \\ -0.6721 & 1.1178 & 0.0321 \\ 0.1415 & -1.3076 & -1.2504 \\ -0.023 & 0.5147 & 0.9828 \\ -0.9873 & 0.3153 & -1.0504 \end{bmatrix}$$

FIGURE 6. V, W network parameter weight adjustments.

The convergence of the PIDNN depends on the choice of the learning step size η, and the learning step size η is calculated according to (6).

$$0<\eta<\frac{1}{\varepsilon^2},(\varepsilon=-\frac{1}{2\sqrt{J}}\frac{\partial J}{\partial W\partial V}). \qquad (6)$$

Here, η is the learning step size; J is the input signal mean square error; V and W are network of the two-level PIDNN. The global-based model has three pairs of inputs and three outputs. The W network is a 6×9 matrix, and the V network is a 9×3 matrix. We set the initial value of V, W, set the number of sampling points $l=50$ according to the backpropagation formula, and set $\eta=0.002$. The weight parameters of the V and W are adjusted according to the weight adjustment function, and 300 iterations are performed. The weights of the V and W networks are then adjusted to obtain a relatively stable PIDNN controller.

Fig. 5 shows the training error of the network. The required accuracy is obtained after 63 iterations. After training, the weight parameters of the V and W are adjusted, as shown in Fig. 6.

C. IMPLEMENTATION FRAMEWORK OF GLOBAL-BASED DVFS ENERGY MANAGEMENT MODEL

To realize the collaborative control of the global-based DVFS energy management model, we constructed a block diagram of the master-slave model, as shown in Fig. 7.

In the master-slave-based model, we use the master GPU run a decoupled neural network in and produce control signals to optimize the energy consumption of the slave GPU. Their communication with the CPU goes through the motherboard system bus. There are two GPUs in the model, and they communicate through the CPU. The description of each functional module in the block diagram is as follows.

1) Master GPU module: This is the master GPU, which runs the global decoupling network and generates three sets of control signal numbers, C_{SM}, C_{ICNT}, and C_{MEM}, for global energy-control of another GPU.
2) Slave GPU module: We call this the controlled GPU. Its energy consumption module also consists of three groups: the SM, MEM, and ICNT modules. The three sets of control signals generated by the master GPU will control the frequency of these modules in real time.

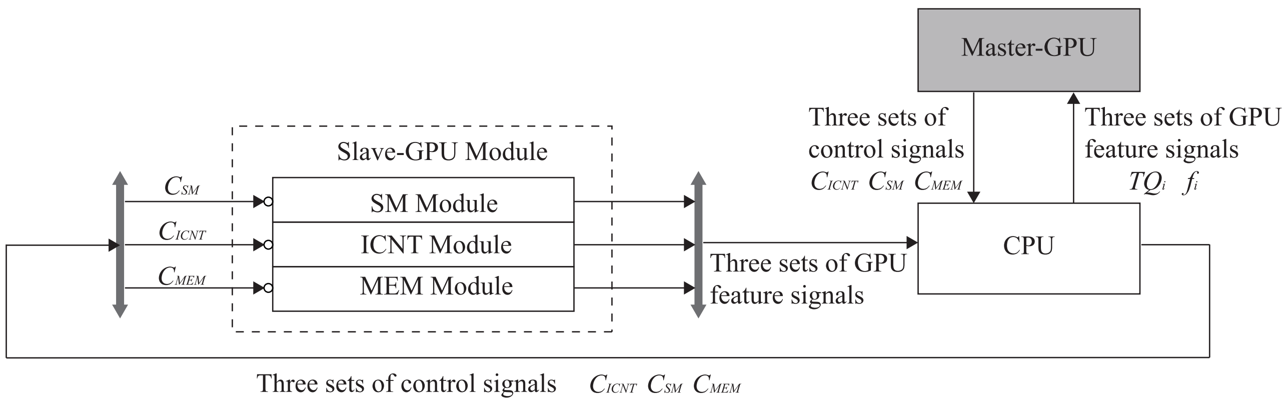
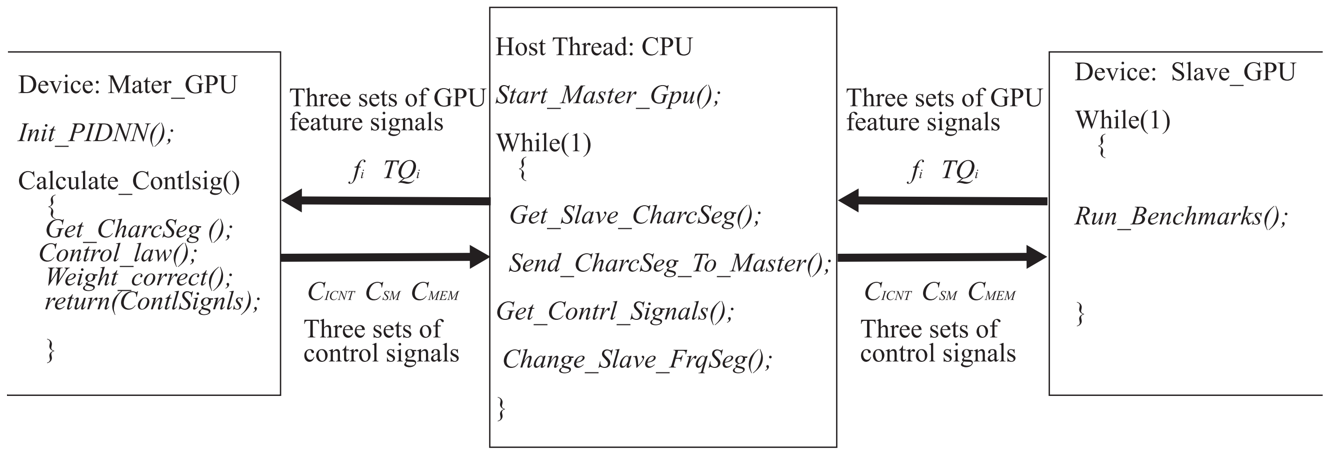

Master-GPU
Three sets of control signals
C_{ICNT} C_{SM} C_{MEM}
Three sets of GPU feature signals
TQ_i f_i
Slave-GPU Module
C_{SM}
C_{ICNT}
C_{MEM}
SM Module
ICNT Module
MEM Module
Three sets of GPU feature signals
CPU
Three sets of control signals C_{ICNT} C_{SM} C_{MEM}

FIGURE 7. Master-slave model energy optimization implementation framework.


Three sets of GPU feature signals
f_i TQ_i
C_{ICNT} C_{SM} C_{MEM}
Three sets of control signals
Three sets of GPU feature signals
f_i TQ_i
C_{ICNT} C_{SM} C_{MEM}
Three sets of control signals

```
Device: Mater_GPU
Init_PIDNN();
Calculate_Contlsig()
  {
   Get_CharcSeg ();
   Control_law();
   Weight_correct();
   return(ContlSignls);
  }
```

```
Host Thread: CPU
Start_Master_Gpu();
While(1)
  {
   Get_Slave_CharcSeg();
   Send_CharcSeg_To_Master();
   Get_Contrl_Signals();
   Change_Slave_FrqSeg();
  }
```

```
Device: Slave_GPU
While(1)
  {
   Run_Benchmarks();
  }
```

FIGURE 8. Global-based-DVF energy management model pseudo-code framework.

3) CPU module: In this model, the CPU is responsible for communication between the master and slave GPUs. It extracts feature signals from the slave GPU, $\{TQ_{SM}, TQ_{MEM}, TQ_{ICNT}$ and $f_{SM}, f_{MEM}, f_{ICNT}\}$, and hands them over to the master GPU, which produces real-time control. The signal, which is responsible for handing over the control signal to the slave GPU, optimizes the power consumption of the slave GPU.

The master-slave model works in three steps. First, the CPU obtains three sets of characteristic signals from the energy consumption modules from the GPU in real time, $\{TQ_{SM}, TQ_{MEM}, TQ_{ICNT}\}$ and $\{f_{SM}, f_{MEM}, f_{ICNT}\}$, and transfers them through the system bus to the master GPU. Second, the master GPU accepts the three sets of characteristic signals, and uses the PIDNN nerve running inside it to generate three sets of control signals,C_{SM}, C_{MEM}, and C_{ICNT}, in real time according to the control law and weight-correction algorithm of the PIDNN network. Third, the CPU acquires three sets of control signals through the function, and the CPU calls the interface to perform real-time control on the operating frequency of the GPU.

D. GLOBAL-BASED DVFS ENERGY MANAGEMENT PSEUDO-CODE FRAMEWORK

We used the energy optimization implementation architecture of the master-slave model to implement an energy optimization strategy based on the NVIDIA CUDA programming model [15]. Communication between the CPU and GPU is performed in master-slave model. Fig. 8 shows a schematic block diagram of the code implementation of the master-slave model, which is used to achieve energy optimization.

The architecture consists of the host thread in the CPU and the devices in the master and slave GPUs. The following three functional modules in Fig. 8 are described.

1) Host Thread: This runs in the CPU, and it has three tasks. The first task is to send a start signal to the Master GPU to inform it to enter the working state. The second task is to be responsible for signal transmission in the two GPUs, which includes obtaining three sets of characteristic signals from the Slave_GPU, obtaining control signals from the Master_GPU, and transmitting them to the Slave_GPU. The third task is to use the obtained control signal to change the

operating frequency of the three energy modules inside the Slave_GPU to achieve energy optimization.

2) Master_GPU Thread: The master GPU process is mainly responsible for initializing the PIDNN neural network and for generating three sets of control signals through calculation.
3) Slave_GPU Thread: This controlled GPU process is mainly responsible for running the benchmark program. It is also responsible for receiving control signals from the CPU and using them to change their working status.

Next, we describe the process of energy optimization, which is completed in three steps.

First, all work starts when the Slave_GPU runs the test program. When the Slave_GPU starts running the process, it will notify the CPU with the function Run_Benchmarks(). The CPU receives the notification and uses Start_Master_GPU() to inform the Master_GPU to start working. The Master_GPU first initializes the global decoupling controller through Init_PIDNN(), and waits for the characteristic signal transmitted by the CPU.

Second, the CPU extracts the feature signal from the inside using the Slave_GPU, and obtains three sets of characteristic signals of the three energy consumption modules of the Slave_GPU through the Get_Slave_CharcSeg() function, $\{TQ_{SM}, TQ_{MEM}, TQ_{ICNT}\}$, and $\{f_{SM}, f_{MEM}, f_{ICNT}\}$. Three sets of characteristic signals are transmitted to the Master_GPU through the Send_CharcSeg_To_Master() function. After receiving the feature signal, the Master_GPU runs the Calculate_Contlsig() method. The method includes calculating the control law of the global decoupling network using the Control_law() function, modifying the parameters with the Weight_correct() function, and returning three sets of control parameters, C_{SM}, C_{ICNT}, and C_{MEM}, with return(ContlSignls).

In the third step, the CPU uses Get_Contrl_Signals() to obtain three sets of control parameters and uses the Change_Slave_FrqSeg() function to change the operating frequency of the Slave_GPU internal energy module in real time.

IV. EXPERIMENT AND ANALYSIS

This section is divided into four parts. The first part describes the experimental design and experimental steps. The second part verifies and analyzes the correlation between the task characteristics and energy consumption. The third part verifies the performance characteristics of the global-based DVFS model. The fourth part verifies the energy consumption characteristics of the global-based DVFS model.

A. EXPERIMENTAL DESIGN AND EXPERIMENTAL ENVIRONMENT

1) EXPERIMENTAL DESIGN

(a) In order to verify the effectiveness of the global-based model, we build an experimental environment that can perform actual energy measurements based on the master–slave model. To achieve energy optimization, we use the Master GPU to run the PIDNN controller to perform coordinated frequency adjustment on the controlled Slave GPU.

FIGURE 9. Experimental environment.

(b) We run the PIDNN controller on the Master GPU. We set the frequency adjustment signal and the controlled signal of the three energy modules once every 1000 clock cycles, taking 50 sampling points per batch. The backward propagation algorithm is used to adjust the network weights of V and W, and the learning step size $\eta = 0.002$. After 64 steps of training and learning, a relatively stable PIDNN controller is obtained.

(c) In order to verify the energy optimization effect of the global-based model, the performance of the Tesla, Fermi, and Kepler GPU platforms are tested, and the results compared. We collect the throughput data and energy consumption data of the three GPUs before and after using the global-based model, and compare and analyze the data.

2) EXPERIMENTAL ENVIRONMENT

In order to verify the effect of the GPU global energy optimization model based on the global-based DVFS model, we set up an experimental environment that can perform actual energy measurements. The environment is shown in Fig. 9.

The experimental components include a Master GPU that runs the PIDNN controller; a frequency-controlled slave GPU; a PC host; and a power meter. A detailed description of the experimental components is as follows.

a: MASTER GPU

An NVIDIA Tesla C2050 [16] was used as a master GPU to run the PIDNN controller in the experiment and generate a frequency adjustment signal for the slave GPU. The specific parameters are described in Table 1.

b: SLAVE GPU

In order to compare and analyze the energy optimization effect of the global-based model, Quadro FX380, GTX480,

TABLE 1. NVIDIA TESLA C2050 experimental parameters.

Hardware Description	Related Parameters
CUDA core quantity	448
CUDA core frequency	1.15 GHz
Double precision floating point performance (peak)	515 Gflops
Single precision floating point performance (peak)	1.03 Tflops
Frame	Fermi
Dedicated memory total capacity	3GB GDDR5
Memory frequency	1.5 GHz
Memory interface	384 bits
Memory bandwidth	144 GB/s
Power consumption	238 W
System interface	PCIe x16 Gen2

TABLE 2. SLAVE GPU experimental parameters (based on different platforms).

	QuadroFX380	GTX460	GTX680
Platform	Tesla	Fermi	Kepler
Main frequency	400 MHz	779 MHz	1006 MHz
Memory capacity	256 MB	1024 MB	2048 MB
Memory type	GDDR2	GDDR5	GDDR5
Memory bandwidth	22.4 GB/S	96.2 GB/S	192.3 GB/S
Bus width	128 bits	192 bits	256 bits
Resolution	1600×1600	1600×1600	2560×1600
Max power	108 W	200 W	195 W

and GTX 680 were selected as the slave GPUs. They were based on the Tesla, Fermi, and Kepler platforms. The module operating frequency within the slave GPU was adjusted by the master GPU. At the same time, we used the slave GPU to run the benchmark and test its throughput and power consumption. Table 2 describes the experimental parameters of the three slave GPUs.

c: PC HOST

Its role is to complete the communication between the FPGA function board and the GPU. The PC host acquires the feature parameters from the GPU and sends the frequency adjustment signal to the FPGA through the I/O interface, thereby optimizing the global energy consumption of the GPU internal modules.

d: TEST PROCEDURE DESCRIPTION

In order to verify the energy optimization effect of the FPGA-based global energy consumption model, we tested the improvement of GPU throughput. The test program was divided into three groups according to the dependency on the cache: high cache sensitivity (HCS), medium cache sensitivity (MCS), and cache insensitivity (CI) [17]. These are described in Table 3.

TABLE 3. Test program classification (classified by cache sensitivity).

High cache sensitivity (HCS)	Inverted Index	IIX
	Page View Count	PVC
	Similarity Score	SSC
	K-means Clustering	KMC
	Page View Rank	PVR
Moderate cache sensitivity (MCS)	String Match	SM
	Needleman-Wunsch	NW
	CFD Solver	CFD
	Fast Fourier Transform	FFT
	Sparse Matrix Vector Multiply	SPMV
Cache-insensitive (CI)	Graphic Diffusion	GD
	Fast Walsh Transform	FWT
	Weather Prediction	WP
	Stencil	STL
	Back Propagation	BP

B. EFFECT OF TASK CHARACTERISTICS ON ENERGY CONSUMPTION

To prove the impact of the task feature on energy consumption, we tested the relationship between the process characteristics (the process access frequency of the cache) and the blocking. We selected 15 test benchmarks [18] and divided the frequency of access to Cache into three groups, according to the program, to verify the relationship between process characteristics and energy consumption.

As can be seen in our experiments, the test program generated much pipeline blocking in the SM. Moreover, this blocking increased as the process increased the frequency of cache access.

From Fig. 10, we find that the higher the dependency of the benchmark program on the cache, the more blocking it will cause. The average blocking value caused by the HCS benchmark was 63.84%, and the average blocking value generated by the MCS benchmark was 43.14%. The average blocking value generated by the CI benchmark program was 11.16%. The process characteristics of cache dependency increased the waiting queues of the L1Cache in SMs, which caused increased blocking of the pipelines, ultimately causing GPU throughput to decrease and operating energy consumption to increase.

Through experiments, we have proved that the process characteristic (cache access frequency) and energy consumption are related, the higher the frequency of access to the process cache the more serious the pipeline blocking, and the higher the corresponding energy consumption.

C. PERFORMANCE OPTIMIZATION CHARACTERISTICS OF THE GLOBAL-BASED DVFS ENERGY MANAGEMENT MODEL

In order to compare the performance of the three Slave GPUs, we used the test procedure outlined in Table 3 to collect throughput rates before and after use of the global-based DVFS model. The test data is shown in Table 4.

In order to more clearly show the performance improvement when using the global-based energy optimization model, we compare the performance of the three Slave GPUs within Fig. 11.

Fig. 11 shows that the throughput of the three slave GPUs improves with the global energy optimization strategy of the global-based model. The GTX680 based on the Kepler platform has the most improvement, and its throughput increases by 13.10% on average, whereas the throughput based on

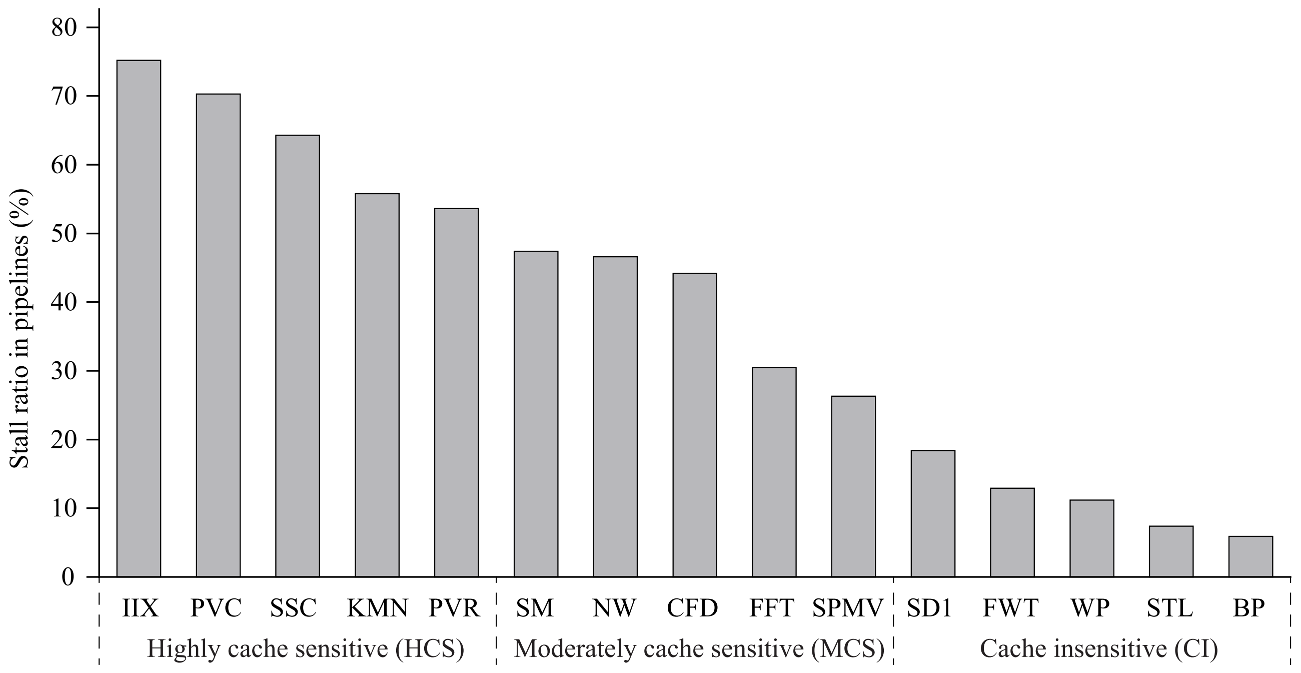


FIGURE 10. Pipeline stall ratio.

TABLE 4. Throughput improvements with the global-based DVFS model.

Cache dependency	BENCHMARK	QuadroFX380			GTX460			GTX 680		
		Throughput Rate of Original Method (MIPS)	Throughput Rate Using Global-Based DVFS Model (MIPS)	Improvement Ratio	Throughput Rate of Original Method (MIPS)	Throughput Rate Using Global-Based DVFS Model (MIPS)	Improvement Ratio	Throughput Rate of Original Method (MIPS)	Throughput Rate Using Global-Based DVFS Model (MIPS)	Improvement Ratio
HCS	IIX	4,471	4923	10.11%	156,836	179,201	14.26%	3,407,645	4,072,833	19.52%
	PVC	4,909	5384	9.67%	168,325	192,185	14.17%	3,651,421	4,278,373	17.17%
	SSC	5,440	5823	7.04%	190,754	209,547	9.85%	3,981,716	4,619,383	16.01%
	MN	6,162	6625	7.52%	215,794	238,573	10.56%	4,532,887	5,237,545	15.55%
	PVR	6,633	7083	6.78%	234,264	255,639	9.12%	4,853,950	5,642,224	16.24%
MCS	SM	7,578	8027	5.92%	254,739	279,564	9.75%	5,311,374	6,163,536	16.04%
	NW	7,926	8284	4.52%	276,558	298,663	7.99%	5,674,597	6,573,472	15.84%
	CFD	8,358	8683	3.89%	305,827	328,052	7.27%	6,232,837	7,122,674	14.28%
	FFT	8,949	9284	3.74%	312,864	331,264	5.88%	6,292,029	7,164,459	13.87%
	SPMV	9,734	10128	4.05%	342,785	353,553	3.14%	6,717,876	7,535,855	12.18%
CI	SD1	11,437	11847	3.59%	365,282	379,755	3.96%	7,215,987	8,032,644	11.32%
	FWT	11,336	11727	3.45%	398,652	415,363	4.19%	7,898,767	8,674,364	9.82%
	WP	12,979	13272	2.26%	428,752	438,325	2.23%	8,387,654	8,997,364	7.27%
	STL	13,026	13198	1.32%	437,289	449,432	2.78%	8,590,878	9,126,575	6.24%
	BP	13,621	13749	0.94%	452,246	460,378	1.80%	8,747,865	9,197,643	5.14%

Fermi GTX460 increases by 7.13% on average. The Quadro FX380 based on the Tesla platform has the least notable improvement (an increase of 4.99%). The experiment shows that the performance of the GPU is significantly improved after using the global energy optimization model of the global-based model, which proves the performance optimization effect of the global-based model. In addition, the experimental results show that the higher the cache dependency of the benchmark program, the more obvious the performance improvement.

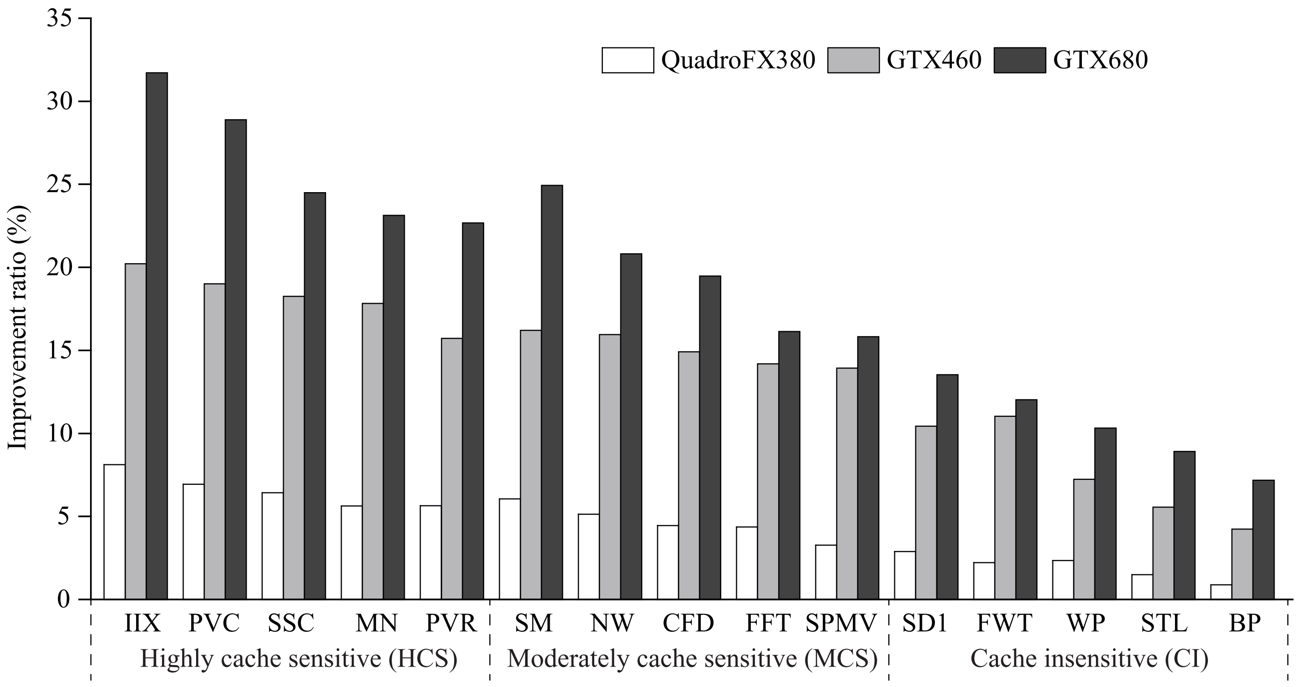


FIGURE 11. Comparison of throughput improvement (based on different platforms).

TABLE 5. Energy consumption with the Global-based DVFS model.

Cache dependency	BENCHMARK	QuadroFX380			GTX460			GTX 680		
		Original Method's Average Power (W)	Average Power Using Global-Based DVFS Model (W)	Impr.	Original Method's Average Power (W)	Average Power Using Global-Based DVFS Model (W)	Impr.	Original Method's Average Power (W)	Average Power Using Global-Based DVFS Model (W)	Impr.
HCS	IIX	27.34	25.12	8.12%	23.25	18.55	20.22%	13.43	9.17	31.72%
	PVC	29.58	27.53	6.93%	24.73	20.03	19.01%	14.26	10.14	28.89%
	SSC	30.63	28.66	6.43%	25.36	20.73	18.26%	14.94	11.28	24.50%
	MN	32.54	30.71	5.62%	26.93	22.13	17.82%	16.82	12.93	23.13%
	PVR	35.96	33.93	5.65%	27.35	23.05	15.72%	17.15	13.26	22.68%
MCS	SM	36.15	33.96	6.06%	29.68	24.87	16.21%	19.37	14.54	24.94%
	NW	39.18	37.17	5.13%	30.73	25.83	15.95%	19.94	15.79	20.81%
	CFD	41.16	39.33	4.45%	31.65	26.93	14.91%	21.26	17.12	19.47%
	FFT	43.82	41.91	4.36%	33.82	29.02	14.19%	23.44	19.66	16.13%
	SPMV	43.98	42.54	3.27%	34.44	29.64	13.94%	24.84	20.91	15.82%
CI	SD1	47.53	46.16	2.88%	36.78	32.94	10.44%	25.62	22.15	13.54%
	FWT	49.23	48.14	2.21%	39.17	34.85	11.03%	28.28	24.88	12.02%
	WP	50.67	49.48	2.35%	39.53	36.67	7.24%	29.36	26.33	10.32%
	STL	53.62	52.82	1.49%	40.84	38.57	5.56%	30.53	27.81	8.91%
	BP	54.59	54.11	0.88%	42.67	40.86	4.24%	31.75	29.47	7.18%

D. ENERGY OPTIMIZATION FEATURES OF GLOBAL-BASED DVFS ENERGY MANAGEMENT MODEL

In order to compare the performance of the three Slave GPUs, we used the test program presented in Table 3 to test the energy consumption of the three Slave GPUs. We collected the energy consumption data before and after use of the global-based DVFS model. The test data is shown in Table 5.

In order to more intuitively display the energy reduction effect of the global-based model, the energy consumption of the three slave GPUs are also presented in Fig. 12.

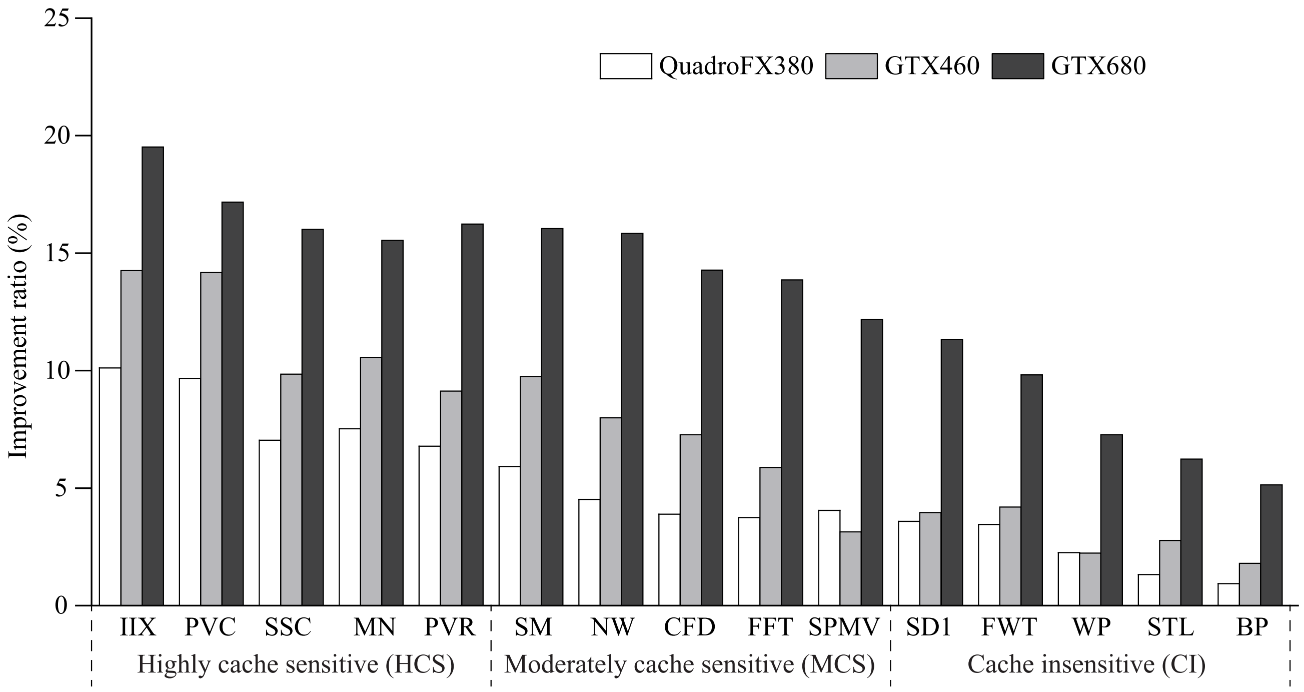


FIGURE 12. **Comparison of energy consumption (based on different platforms).**

This figure clearly shows that the energy consumption of the three slave GPUs decreases after using the global-based model's global energy optimization strategy. Among them, the energy loss of GTX680 based on the Kepler platform is the most obvious: its energy consumption drops by 18.67% on average. Meanwhile, the energy consumption based on Fermi GTX460 decreases by 13.65% on average, and the power consumption of Quadro FX380 based on the Tesla platform drops by 4.39% on average. This shows that after using the global-based model's global energy optimization model, the GPU's energy consumption has a significant decline, which proves the energy efficiency of the global-based model. In addition, the experimental results show that the higher the cache dependency of the benchmark program, the more obvious the performance improvement.

Combining the Slave GPU performance improvement and energy optimization results, we can see that the global-based model has improved performance and energy optimization for GPUs on all three platforms. The GTX680 based on the Kepler platform has the most obvious improvement, followed by the GTX460 based on the Fermi platform. Meanwhile, the Quadro FX380 based on the Tesla platform has the least obvious improvement.

The energy optimization achieved by the global-based model is based on the following two points. First, the PIDN controller is used to decouple the three energy consumption modules in the GPU at the same time, and the operating frequency of the three energy consumption modules is coordinated. Second, the global-based model has a time advantage in parameter adjustment of the PIDNN network, and can coordinate the operating frequency of the energy consumption module in the GPU.

V. CONCLUSIONS

In this paper, a new DVFS method is used to control the energy consumption of a GPU. This approach uses two energy-related strategies. The first strategy is that the energy management method adaptively adjusts the energy consumption module of the GPU based on the task feature. We have experimentally proven that this strategy is effective for GPU energy management. The second strategy is to use PIDNN to achieve coordinated adjustment of GPU multi-energy modules. We built the global-based DVFS energy management model to achieve global GPU energy optimization. This global-based DVFS method improves both GPU performance and energy efficiency on different platforms.

We have only implemented the global-based DVFS energy management model in a single GPU and have not considered applying this strategy in a multi-GPU cluster environment. Looking ahead, to solve high-performance computing problems, GPU-based computing systems will move toward multi-core, multi-card clusters. This means that a GPU workstation will often contain multiple GPU computing cards. We envisage applying the global-based DVFS model energy optimization method to the GPU cluster.

ACKNOWLEDGMENT

We thank LetPub (www.letpub.com) for its linguistic assistance during the preparation of this manuscript.

REFERENCES

[1] X. Mei, L. S. Yung, K. Zhao, and X. Chu, "A measurement study of GPU DVFS on energy conservation," in *Proc. Workshop Power-Aware Comput. Syst.*, Farmington, CT, USA, Nov. 2013, p. 10. doi: 10.1145/2525526.2525852.

[2] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proc. 8th Int. Symp. High Perform. Comput. Archit.*, Cambridge, MA, USA, Feb. 2002, pp. 29–40. doi: 10.1109/HPCA.2002.995696.

[3] X. Mei, Q. Wang, and X. Chu, "A survey and measurement study of GPU DVFS on energy conservation," *Digit. Commun. Netw.*, vol. 3, no. 2, pp. 89–100, May 2017. doi: 10.1016/j.dcan.2016.10.001.

[4] D. H. K. Kim, C. Imes, and H. Hoffmann, "Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics," in *Proc. IEEE 3rd Int. Conf. Cyber-Phys. Syst. Netw. Appl.*, Aug. 2015, pp. 78–85.

[5] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and performance characterization and modeling of GPU-accelerated systems," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.*, May 2014, pp. 113–122.

[6] Y. Liang, H. P. Huynh, K. Rupnow, R. S. M. Goh, and D. Chen, "Efficient GPU spatial-temporal multitasking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 748–760, Mar. 2015. doi: 10.1109/tpds.2014.2313342.

[7] S. Song and K. W. Cameron, "System-level power-performance efficiency modeling for emergent GPU architectures," in *Proc. 21st Int. Conf. Parallel Archit. Compilation Techn.*, Minneapolis, MN, USA, Sep. 2012, p. 473. doi: 10.1145/2370816.2370903.

[8] K. Dev, X. Zhan, and S. Reda, "Power-aware characterization and mapping of workloads on CPU-GPU processors," in *Proc. IEEE Int. Symp. Workload Characterization*, Providence, RI, USA, Sep. 2016, pp. 1–2. doi: 10.1109/IISWC.2016.7581285.

[9] G. Tang, W. Jiang, Z. Xu, F. Liu, and K. Wu, "Zero-cost, fine-grained power monitoring of datacenters using non-intrusive power disaggregation," in *Proc. 16th Annu. Middleware Conf.*, Vancouver, BC, Canada, Dec. 2015, pp. 271–282. doi: 10.1145/2814576.2814728.

[10] K. Li, "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr./Jun. 2016. doi: 10.1109/tcc.2015.2440238.

[11] J. Li, B. Guo, Y. Shen, D. Li, and Y. Huang, "A modeling approach for energy saving based on GA-BP neural network," *J. Elect. Eng. Technol.*, vol. 11, no. 5, pp. 1289–1298, Sep. 2016. doi: 10.5370/jeet.2016.11.5.1289.

[12] H. Shu, *PID Neural Network and Its Control System*. Beijing, China: National Defense Industry Press, 2006.

[13] C. Liu and H. Song, "Design of electric loading system in flight simulator based on PIDNN," in *Proc. Int. Conf. Mechatronic Sci. Electr. Eng. Comput.*, Jilin, China, Aug. 2011, pp. 2623–2626. doi: 10.1109/mec.2011.6026030.

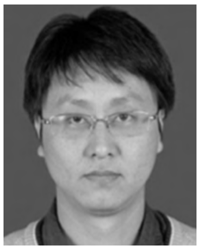
[14] Y. Zhang, H.-B. Zhu, and T.-Q. Lu, "PIDNN decoupling control of boiler combustion system based on MCS," in *Proc. 29th Chin. Control Decis. Conf.*, Chongqing, China, May 2017, pp. 7110–7115. doi: 10.1109/ccdc.2017.7978466.

[15] *NVIDIA's Next Generation CUDA Compute Architecture: Fermi*, NVIDIA, Santa Clara, CA, USA, 2016.

[16] *TESLA C2050/C2070 GPU Computing Processor Supercomputingat 1/10th the Cost*. Accessed: 2010. [Online]. Available: https://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

[17] J. A. Stratton *et al.*, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," Center Reliable High-Perform. Comput., 2012. [Online]. Available: http://impact.crhc.illinois.edu/Shared/Docs/impact-12-01.parboil.pdf

[18] J. Coplin and M. Burtscher, "Energy, power, and performance characterization of GPGPU benchmark programs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, Chicago, IL, USA, May 2016, pp. 1190–1199. doi: 10.1109/ipdpsw.2016.164.



YANHUI HUANG received the B.S. and M.S. degrees in radio electronics and computer science from Sichuan University, in 1997 and 2002, respectively, where he is currently a Lecturer with the School of Computer Science. His current research interests include embedded real-time systems and green computing.

BING GUO received the B.S. degree in computer science from the Beijing Institute of Technology in China, in 1991, and the M.S. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China, China, in 1999 and 2002, respectively. He is currently a Professor with the School of Computer Science, Sichuan University, China. His current research interests include embedded real-time systems and green computing.

YAN SHEN received the M.S. degree in mechatronics engineering and the Ph.D. degree in measuring and testing technology and instruments from the University of Electronic Science and Technology of China, in 2001 and 2004, respectively. She is currently a Professor with the Control Engineering College, Chengdu University of Information and Technology. Her main research interests include distributed measurement systems and embedded system development.