IOP Conf. Series: Journal of Physics: Conf. Series **1487** (2020) 012028 doi:10.1088/1742-6596/1487/1/012028

Optimization of Energy Efficiency for FPGA-Based Convolutional Neural Networks Accelerator

Yongming Tang[1,2,a], Rongshi Dai[1,b] and Yi Xie[2]

[1]Joint International Research Laboratory of Information Display and Visualization,
Southeast University, Nanjing, 211189, China
[2]Science and Technology on Electro-optic Control Laboratory, Luoyang, China

[a]tym@seu.edu.cn, [b]drsseu@outlook.com

Abstract. Convolutional neural network (CNN) is widely applied to image recognition with high recognition accuracy. CNN has a wider implementation in general-purpose processors and can be accelerated on FPGA. CNN has a unique way of computing, but general-purpose processors are not efficient for CNN and cannot meet energy efficiency requirements. And the previous studies on FPGA did not involve an energy-efficient implementation on FPGA. We innovatively propose energy efficiency models and implement high energy efficiency CNN on FPGA. We implemented the LeNet-5 network model on the GENESYS 2 board and compared it to the traditional processor and previous studies. By comparison, the computing throughput of CPU, GPU and FPGA are 3.831GFLOPS, 27.143GFLOPS and 19.61GFLOPS respectively, and their powers are 32.15W, 52W, 4.152W respectively. The final energy efficiency (GFLOPS/W) is 0.119GFLOPS/W, 0.522 GFLOPS/W, 4.723 GFLOPS/W, so the energy efficiency of FPGA are far superior to that of CPU and GPU. Since the energy efficiency we achieved on FPGA is also higher than that of FPL2009 and FPGA2015, and we have achieved good experimental results in energy efficiency.

1. Introduction

In recent years, convolutional neural network (CNN) has made great contributions in different fields. As a widely used neural network, it has great influence in the fields of image recognition, image search and image classification [1]. Since CNN is inspired by the behavior of optic nerves in living creatures, it uses the convolution kernel to extract the features and through the mapping relationship between the neural layers. Finally, it transforms the features into the results for output, and has a high accuracy, high usage and ease of implementation.

In the implementation of the hardware platform of CNN, most of them are implemented by CPU and GPU. However, for some tasks, the front-end platform needs to have small size and low power consumption. Traditional processors are less energy efficient than we expected, so we need to find a new front-end platform to complete the task. Therefore, implementations on FPGA and ASIC chip are achievable platforms [2] [3]. Because FPGA and ASIC chips as accelerators to implement CNN are not dependent on the system, directly related to data stream processing, and have better performance in resource utilization. However, ASIC chips have high manufacturing cost and long development cycle, so FPGA is a suitable hardware implementation platform in a short time. Due to the advantages of FPGA development, such as high performance, high energy efficiency and short development cycle, FPGA-based CNN accelerators have attracted more and more researchers' attention [1] [2] [4][5] [6].

This article initially implemented the LeNet-5 [7] model on the GENESYS 2 board. While making reasonable use of FPGA's resources, improve energy efficiency as much as possible [6]. Considering the different power consumption of different FPGA's resources, the rational allocation of resources to reduce power consumption has greatly improved resource utilization and improved system energy efficiency to some extent.

The main contributions of this work are summarized as follows.

- We propose a series of schemes for accelerating convolutional layer operations for data bandwidth and execution sequence logic, and surpass the performance of CPU and get close to the that of GPU in terms of acceleration performance.
- In the energy-efficient design, considering the acceleration effect and power consumption change, an optimization model for network layer and resource allocation is proposed.
- We implemented an energy-efficient CNN accelerator with an energy efficiency of 4.73 GFLOPS / W, which is higher than the energy efficiency of FPL2009 [1] and FPGA2015 [8]. To the best of our knowledge, this achieves the current high energy efficiency CNN accelerator.

The rest of this article is organized as follows: Section 2 introduces the background of CNN and analyzes the factors of system energy efficiency. Section 3 provides optimization model and describes the details of an energy-efficient accelerator implementation. Section 4 shows the results of our experiments. Section 5 summarizes the paper.

2. Background

2.1. CNN Basics

Over the past decade, there has been a significant increase in software strength and hardware performance, such as the development of deep learning theory and the update iteration of supercomputers, CNN has developed rapidly and are widely used in computer vision and natural language processing and other fields [9]. Since CNN is inspired by the behavior of optic nerves in living creatures. The parameter sharing in the hidden layer of CNN and the sparse connection of neurons between different neural layers ensure that CNN can simplify the data of the input layer with a relatively small amount of computation, which is also convenient for the latter feature extraction. When CNN is used for supervised learning, its feedforward network part is mostly used for image recognition and classification, and the feedback part is used for training. By using the already trained network weight data, we need to implement its feedforward network part on FPGA.

The main CNN framework consists of two main parts: feature extractor and classifier. The function of the feature extractor is to extract the features of the input image and map them to the subsequent feature maps through the feature map. The features of these images are not unique, and mainly was gotten through the sliding process of the convolution kernel. And the classifier function is the process of transforming the feature layer into an output structure.

Taking LeNet-5 as an example, the feature extractor includes a convolutional layer and a downsampling layer, and the fully connected layer is a classifier. The weights and bias parameters of the LeNet-5 network model are shown in table 1.

Table 1. Weights and bias of the LeNet-5 network model.

Layer	Conv1	Conv2	Conv3	Fullconnect1	Fullconnect2
Weights[a]	<1,5,5,6>	<6,5,5,16>	<16,5,5,120>	<120,84>	<84,10>
Bias[b]	6	16	120	84	10

[a] Conv: <input layer number, convolution kernel length, convolution kernel width, output layer number>, Fullconnect: <input layer number, output layer number>.

[b] The number of bias is the number of output layers.

2.2. Performance limit of FPGA

The dominant frequency of FPGA is usually several hundred MHz, while the frequency of a generic processor can be up to GHz. How can the speed of data processing of FPGA be greatly improved?

First, increase the number of parallel calculations. It is necessary to properly design the parallel development in consideration of the total amount of resources, which is one of the most effective methods for speed improvement. Second, increase the number of ports in the memory of variables. Even if the calculation process is expanded, for the data in the memory, each clock can only be read and written once, which affects the overall speed. Therefore, it is necessary to expand the memory port. Then, pipeline operation. By using the pipeline operation, the latter operation can start at the middle time from the previous operation, reducing the waiting time and improving the system work efficiency. Finally, change the order in which convolution calculations are performed. By reasonably changing the logical order, the time crowding effect between data streams can be changed.

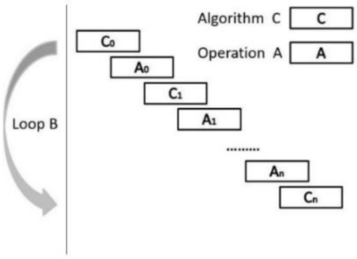


Figure 1. We insert operation A in the space of algorithm C, avoiding the waste of idle waiting time in the loop.

For example, as shown in figure 1, in a loop B, we will implement algorithm C. Algorithm C needs to perform repeated reading and writing address operations on the same address, which will cause timing congestion. If you can change the steps of the calculation, you can put some operations that can change the address of this variable, such as the operation A, into this loop B. Operation A has been added between the two algorithm C, so the operation A is completed in the waiting time between the two algorithm C. In this way, the acceleration of a data processing is completed without changing the resources of the system.

2.3. Power consumption

As can be seen from 2.2, the lower dominant frequency of FPGA also reduces the system power consumption of FPGA, which is the most important factor for FPGA to achieve low power consumption. The power consumption of FPGA can be divided into static power and dynamic power. This is mainly because FPGA is made up of transistors, and the transistors also have static power and dynamic power. So which power consumption can be reduced or avoided, in addition to the power generated by the inherent properties of the device?

FPGA resources are divided into the following types: LUT, FF, BRAM, DSP, IO, MMCM, etc. Power consumption can be reduced by rational use of resources. BRAM has the largest power consumption. The smaller the BRAM block is, the smaller the power consumption is, but the logic and layout become larger. The MMCM consumes a lot of power, so it is necessary to reduce the amount of usage as much as possible. A single LUT, FF, and DSP consume less power, but the larger number of LUTs is inevitable and is an essential part of the resource.

3. Accelerator design exploration

This section will first introduce an overview of our accelerator architecture and illustrate several design challenges on the FPGA platform. In order to overcome these challenges, we have proposed corresponding optimization methods, such as optimization model, HLS optimization, low power optimization and design space exploration.

3.1. Optimization model

3.1.1. Partial energy efficiency optimization model. When FPGA implement energy-efficient CNN, we need to consider both accelerated performance and power consumption. Since the optimization of the CNN is optimized according to the network layer, it is necessary to consider whether partial energy efficiency optimization promotes the optimization of system energy efficiency. We propose the **Partial energy efficiency optimization model** algorithm, according to which we can determine whether partial optimization can improve the energy efficiency of the system.

We define a network layer **a**. The other layers are collectively called **b**. The network delay is defined as **t**. We define the speed as $1/(t_a+t_b)$, which indicates the speed of completing the unit task. The power consumption is **P**, so the system energy efficiency E_1 is as follows.

$$E_1 = \frac{\frac{1}{t_a+t_b}}{P_a+P_b} \tag{1}$$

Let $\frac{t_b}{t_a} = \mu_t, \frac{P_b}{P_a} = \mu_P, Q_a = t_a * P_a, Q_a$represent the energy consumed by the unit task.

$$E_1 = \frac{1}{Q_a}\frac{\frac{1}{1+\mu_t}}{1+\mu_P} \tag{2}$$

We optimize network layer a, which increases α (>1) times in speed, and at the same time increases power consumption by β (>1) times. The optimized E_2 is as follows.

$$E_2 = \frac{\frac{1}{t_a+\frac{t_b}{\alpha}}}{P_a+\beta P_b} \tag{3}$$

$$E_2 = \frac{1}{Q_a}\frac{\frac{1}{1+\frac{\mu_t}{\alpha}}}{1+\beta\mu_P} \tag{4}$$

$$\frac{E_2}{E_1} = \frac{(1+\mu_t)(1+\mu_P)}{(1+\frac{\mu_t}{\alpha})(1+\beta\mu_P)} \tag{5}$$

Because we need $\frac{E_2}{E_1} > 1$, so

$$\mu_t + \mu_P + \mu_t\mu_P > \frac{\mu_t}{\alpha} + \beta\mu_P + \frac{\beta}{\alpha}\mu_t\mu_P \tag{6}$$

By analyzing the situation of the above formula, it is divided into the following cases.

- **$\alpha = \beta$**

  Get the following.

  $$\mu_t + \mu_P > \frac{\mu_t}{\alpha} + \alpha\mu_P \tag{7}$$

  If $\mu_t < \mu_P$, we get $\frac{\mu_t}{\mu_P} < \alpha < 1$, and because $\alpha>1$, it does not exist.

  If $\mu_t > \mu_P$, we get $1 < \alpha < \frac{\mu_t}{\mu_P}$.

- **$\alpha>\beta$**

  If $\alpha>\beta$, then a sufficient and unnecessary condition of the above formula is as follows.

  $$\mu_t + \mu_P > \frac{\mu_t}{\alpha} + \beta\mu_P \tag{8}$$

  which is equivalent to

  $$\frac{\mu_P}{\mu_t} < \frac{\alpha-1}{\alpha(\beta-1)} \tag{9}$$

  and $\alpha > \beta > 1$, so we get $\frac{\mu_P}{\mu_t} < 1$.

  Therefore, when $\alpha>\beta$ and $\frac{\mu_P}{\mu_t} < 1$, it is possible to make $E_1 < E_2$.

- **$\alpha<\beta$**

  If $\alpha<\beta$ and $\frac{\mu_P}{\mu_t} > 1$, there must be $E_1>E_2$.

In summary, when $1 < \beta \leq \alpha < \frac{\mu_t}{\mu_P}$, the system energy efficiency will increase.

3.1.2. Global energy efficiency model. Similarly, we can get the **Global energy efficiency model** by the **Partial energy efficiency optimization model**. Since the energy consumption of the system is related to the type of resources, the power consumption of each resource is different. We take DSP as an example for analysis. DSP is the resource used by hardware multiplication. In general, if the DSP resources are doubled, the system speed v can be doubled. Before optimization, the energy efficiency is E_1, the power consumption of the DSP is P_2, and the other power consumption is P_1. If the optimized DSP resource is N times the previous resource, P_2 will also increase by N times. Due to the increase of DSP resources, the use of resources such as P_1 will be correspondingly improved by M times, which is less than N times.

$$E_1 = \frac{v}{P_1 + P_2} \tag{10}$$

$$E_2 = \frac{v'}{P_1' + P_2'} = \frac{N * v}{MP_1 + NP_2} = \frac{v}{{MP_1}/{N} + P_2} > E_1 \tag{11}$$

3.2. HLS optimization

Xilinx Vivado® HLS converts C to Register Transfer Level (RTL) and can be integrated into Xilinx Field Programmable Logic Arrays.

In the feedforward computation perspective, a previous study [10] proved that convolution operations will occupy over 90% of the computation time. So in this work, we will focus on accelerating convolutional layers [11]. For input images of multi-feature, convolution operations usually have six layers of loops: input layer, output layer, image line, image column, convolution kernel row, and convolution kernel column. Therefore, when a convolution operation is performed by a generic processor, it takes a lot of time to perform a convolution operation.

```
for (row=0; row<R; row++) {
   for (col=0; col<C; col++) {
      for (ti=0; ti<N; ti++) {
         for (to=0; to<M; to++) {
            for (i=0; i<K; i++) {
               for (j=0; j<K; j++) {
                  image_out[to][row][col] +=
            (image_in[ti][row+i] [col+j]
            *wconv[ti][to][i][j]);
}}}}}}
```

Code 1. Unoptimized convolution operation

```
for (i=0; i<K; i++) {
   for (j=0; j<K; j++) {
      for (row=0; row<R; row++) {
         for (col=0; col<C; col++) {
#pragma HLS PIPELINE
            for (ti=0; ti<N; ti++) {
               for (to=0; to<M; to++) {
                  image_out[to][row][col] +=
            (image_in[ti][row+i] [col+j]
            *wconv[ti][to][i][j]);
}}}}}}
```

Code 2. Optimized convolution operation

$$\mathrm{image\_out[to][row][col]} {+}{=} (\mathrm{image\_in[ti][row + i][col + j]} * \mathrm{wconv[ti][to][i][j]}) \tag{12}$$

The entire calculation process of the convolution process is completed by formula 12, and the multiplication and cumulative sum operations of the entire convolution process are completed.

Taking convolutional layer 2 as an example, when K=5, R=10, C=10, M=16, and N=6, the speed of code 2 can be increased to 509 times of code 1.

3.2.1. Increase the number of parallel calculations. The system needs to plan more resources for the underlying routing planning, and will also need to add more ports for data transfer to match the highly parallel data flow.

3.2.2. Increase the number of ports in the memory of variables. The system speed can be greatly improved by increasing the number of parallel systems, but it is necessary to read and write a variable multiple times at the same time. This variable is required for Partition operation, which will require more BRAM.

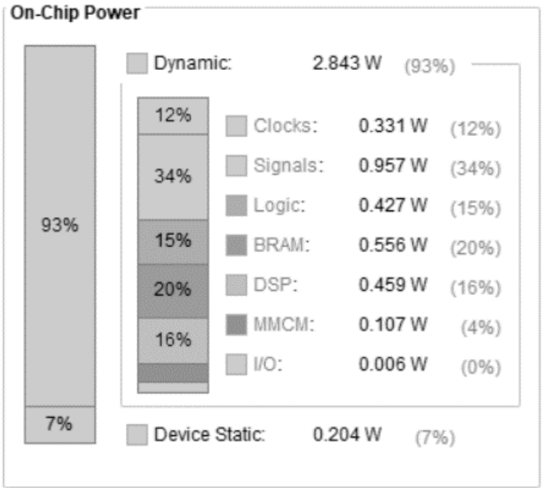
3.2.3. Pipeline operation. Multiple operations can be completed in one operating clock cycle. The time between each operation is called Initiation Interval and is particularly important. If you use the pipeline, Initiation Interval can be shortened to a minimum of one clock cycle, which has obvious effect on the system acceleration effect.

3.2.4. Change the order in which convolution calculations are performed. By comparing the optimized code with the unoptimized code, due to the Pipeline optimization, we need to put the loop of the input layer and the number of output layers (the loop of length M and N) into the innermost layer. If the convolution kernel is cycled into the outer layer, the inner loop will not be limited to the address, so the next operation does not need to be performed after the previous operation is completed. This optimization greatly increases the speed of the system when resources are constant [12].

Taking the Conv2 layer of our implementation as an example, it can be seen from the comparison test that this optimization measure increases the speed of the Conv2 by 9.5 times.

3.3. Low power optimization

The final board power is 4.152W. As shown in figure 2, in the Vivado software simulation, the FPGA chip consumes 3.048W, of which the static power consumption is 0.201W and the dynamic power consumption is 2.843W. The main part of dynamic power consumption is Signals and BRAM, with 34% and 20% power consumption respectively.



Dynamic: 2.843 W (93%)
Clocks: 0.331 W (12%)
Signals: 0.957 W (34%)
Logic: 0.427 W (15%)
BRAM: 0.556 W (20%)
DSP: 0.459 W (16%)
MMCM: 0.107 W (4%)
I/O: 0.006 W (0%)
Device Static: 0.204 W (7%)

Figure 2. The power consumption profile of each part of the system simulation. In this figure, we can find that the loss of each part is different, and the power consumption of Signals is the largest. (Vivado 2018.2)

From the content of figure 2, we can see the power consumption of each part. The system clock power is only 0.331W, and it is unchangeable power consumption, which is an inherent property of the system. Signals are divided into three categories, Data, Clock Enable and Set/Reset, which are 0.939W, 0.012W and 0.005W, respectively, so Data of Signals is the most pivotal part of power consumption. The system logic module only occupies 0.427W and has a low power consumption. The second is that the BRAM power consumption is 0.556W. Because the data amount of the variable is a fixed size, the amount of data used by the BRAM is constant. After the Partition operation, the BRAM is split into small blocks and the power consumption is reduced, but more decoding logic is used. Therefore, the power consumption changes before and after BRAM partitioning is small. Once the network model is determined, the BRAM power consumption is difficult to optimize.

The use of DSP will be related to the speed of the system. If the DSP usage is increased by N times, the power consumption will be increased by about N times, and the effect of acceleration will be increased by about N times. From formula 11, we can figure out whether DSP optimization leads to an increase in energy efficiency. And from figure 2, we can know P1=2.59W, P2=0.46W.

$$\frac{E_2}{E_1}=\frac{2.59+0.46}{\frac{2.59*\mathrm{M}}{N}+0.46}=\frac{6.6*N}{5.6\mathrm{M}+N}>1 \tag{13}$$

MMCM usage is related to clock usage and is inherent to the system and cannot be changed for optimization. I/O is almost unused and consumes less power.

After considering the energy consumption of all aspects of the resource, the overall energy efficiency can be improved by optimizing the number of parallel and pipeline operations, so that the speed and power consumption are balanced to achieve optimal energy efficiency.

3.4. Design space exploration

The energy efficiency optimization of a CNN accelerators on FPGA is mainly achieved by rational application of FPGA's resources and HLS optimization measures. After the neural network model is determined, it is necessary to select an appropriate FPGA chip for the experiment. By using the HLS application tool, acceleration can be performed with little or no increase in resources, and the system energy efficiency will be greatly improved.

Taking FPGA2015 Cheng's paper as an example, it uses a Virtex 7 board to implement CNN with a throughput of 1.33GFLOP, but Cheng's energy efficiency is lower than ours. The resources we use are lower than Cheng's, so the final energy efficiency results do not necessarily depend on the resource capacity of the hardware platform. We need to choose the right hardware platform based on the throughput of the network model.

After the optimization measures, the system energy efficiency is improved by using Vivado HLS. Through the comprehensive comparison of the optimized speed and resources, the law of energy efficiency changes is obtained.

Table 2. The number of delay clock of each layer of the network is as follows.

Lays	*Unoptimized*	*Optimized*	*Multiple*
Lay1	1,317,697	2986	441x
Lay2	8653	199	43x
Lay3	2,692,143	5291	509x
Lay4	2861	186	15x
Lay5	486,902	3912	124x
Lay6	101,463	1889	54x
Lay7	8590	788	11x
Total	4,620,456	18,409	251x

Table 3. The number of resource utilization of the CNN module is as follows.

Resources	*Optimized*	*Unoptimized*	*Multiple*
BRAM	829	274	3.0x
DSP	623	25	24.9x
FF	110,979	5467	20.3x
LUT	129,493	8858	14.6x

As can be seen from table 2 and table 3, we compare the acceleration effects and resource usage before and after optimization in detail. In the optimization of system acceleration, the final improvement was 251 times, and the maximum utilization factor of resources was increased by 24.9 times (DSP). The power consumption of the system is often positively related to its resource usage, and the system energy efficiency is increased by at least 10.1 times. Therefore, the appropriate HLS optimization measures will also improve the system energy efficiency while improving the speed.

4. Evaluation

First, this section will introduce our experimental environment settings, and then provide a detailed comparison of experimental results and related experimental results.

4.1. Experimental Setup

The accelerator design was implemented by Vivado HLS (v2018.2), which allows compiling in C and exporting RTL to an IP core. First, the simulation before the rapid synthesis is completed using the C simulation and C/RTL co-simulation of the Vivado HLS. Then the exported RTL is synthesized and implemented using Vivado v2018.2. In the end, our implementation is based on the GENESYS 2 board, and the core chip is the Xilinx FPGA xc7k325tffg900. The set operating frequency is 100MHz, and the Vivado software runs on a PC of Intel(R) Core(TM) i5-4590 CPU@3.30GHz.

4.2. *Experimental result*

In this section we will report on the resource usage and then compare the energy efficiency of the software implementation (on the CPU and GPU) and our accelerator implementation (on FPGA). Finally, we give a comparison between our implementation of CNN and others to implement CNN on FPGA.

The internal layout of FPGA is provided by the Vivado toolset and it can report resource usage. As shown in the table 4, it can be seen that our CNN accelerator has almost fully utilized the hardware resources of FPGA.

Table 4. FPGA Resource Utilizaiton.

	DSP	*BRAM*	*LUT*	*FF*
Used	623	373.5	85032	69565
Available	840	445	203,800	47,600
Utilization（%）	74.17	83.93	41.72	17.07

The specific use effect on system resources is as shown in the table 4, and energy efficiency is optimized by rational allocation of resources.

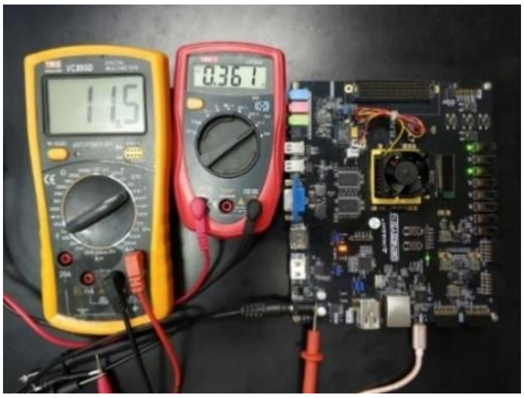


Figure 3. Power measurement of on-board execution

The power consumption we have achieved is shown in figure 3.

Table 5. Comparison to CPU/GPU.

	CPU (i5-4590)	GPU(GTX 1080ti)	FPGA（G2）
Performance(GFLOPS)	3.831	27.143	19.61
Power(W)	32.15	52	4.152
Energy Efficiency(GFLOPS/W)	0.119	0.522	4.723

We have implemented the LeNet-5 model on CPU and GPU. The network models are the same on all three platforms. We develop CNN on the CPU and GPU with some common tools to use resources as efficiently as possible. Since their development environment is done on Windows systems, dynamic power consumption is considered as system power consumption, which reduces the impact of static power consumption. However, the power consumption of FPGA is the total power consumption of the system, which is more practical in the comparison of energy efficiency. As can be seen from the table 5, the energy efficiency on FPGA is much higher than that on the CPU and GPU, and its throughput rate is 19.61 GFLOPS, which is closer to the throughput of the GPU. The performance of acceleration and energy efficiency on FPGA achieve the expected results, and have a good realization significance.

Table 6. Comparison to previous implementations.

	FPL2009	FPGA2015	Our Impl.
Frequency(MHz)	125	100	100
FPGA chip	Virtex4 SX35	Virtex7 VX485T	GENESYS 2
FPGA capacity	126 DSP 23,872 slices	2800 DSP 75,900 slices	840 DSP 50,950 slices
Performance(GOPS)	5.25	61.62	19.61
Power(W)	15	18.6	4.15
Energy Efficiency(GOPS/W)	0.35	3.31	4.73

Compared with Cheng's experimental results (FPGA2015), the implementation of this paper has higher energy efficiency and is implemented on a system platform with relatively low resources. Therefore, there is no absolute relationship between system energy efficiency and system resources, but it has a great relationship with network model size and optimization scheme. We achieved 4.15W of power consumption on the GENESYS2 board and achieved new breakthroughs in energy efficiency with lower throughput. In terms of system energy efficiency, we can not only care about the acceleration effect of the system, but also optimize the system energy consumption through appropriate methods. As an energy-efficient solution, acceleration and power usage are places where we need to do more in-depth research in the future.

5. Conclusion

In this paper, we present an energy-efficient solution for FPGA-based CNN and achieve relatively outstanding energy performance. By reasonably analyzing the relationship between the acceleration performance and the energy consumption of FPGA, we propose two optimization models and finally achieve better energy efficiency. By comparing the performance with CPU and GPU, we can find that our solution has an absolute advantage in the optimization of energy efficiency of FPGA. The energy efficiency of FPGA has far exceeded the energy efficiency of the CPU and GPU. Finally, we realize an energy efficiency on GENESYS 2 board which outperforms previous work.

Acknowledgement

The program is financially supported by Science and Technology on Electro-optic Control Laboratory and Aeronautical Science Foundation of China (20155169017).

References

[1] Farabet C, Poulet C, Han J Y, et al 2009 *International Conference on Field Programmable Logic & Applications* (Czech Republic: Prague) pp 32-37

[2] Chakradhar S T, Sankaradass M, Jakkula V, et al 2010 *International Symposium on Computer Architecture* (France: Saint-Malo) pp 19-23

[3] Boutros A, Yazdanshenas S, Betz V 2018 *ACM Transactions on Reconfigurable Technology and Systems* vol 11 pp 1-23

[4] Qiu J, Wang J, Yao S, et al 2016 *International Symposium on Field-Programmable Gate Arrays* (Beijing: Tsinghua National Laboratory for Information Science and Technology) pp 26-35

[5] Venieris S I, Bouganis C S 2016 *International Symposium on Field-programmable Custom Computing Machines* (UK: Department of Electrical and Electronic Engineering) pp 40-47

[6] Peemen M, Setio A. A. A, Mesman B, et al 2013 *International Conference on Computer Design* (Netherlands: Eindhoven University of Technology)

[7] Lauer F, Suen C Y, Gérard Bloch 2007 *Pattern Recognition* vol 40 pp 1816-1824.

[8] Zhang C, Li P, Sun G, et al 2015 *International Symposium on Field-programmable Gate Arrays* (Peking University: Center for Energy-effcient Computing and Applications)
[9] Bei H, Rui A, Yang Y, et al 2016 *Intelligent Vehicles Symposium* (China: Baidu Map)
[10] J. Cong and B. Xiao 2014 *International Conference on Artificial Neural Networks* (University of California: Computer Science Department) pp 281-290
[11] Ma Y, Cao Y, Vrudhula S, et al 2017 *International Symposium on Field-programmable Gate Arrays* (School of Electrical, Computer and Energy Engineering) pp 45-54
[12] L.-N. Pouchet, P. Zhang, P. Sadayappan, and J. Cong 2013 *International Symposium on Field Programmable Gate Arrays* (USA: University of California Los Angeles) pp 29-38,