An energy-efficient coarse grained spatial architecture for convolutional neural networks AlexNet

Boya Zhao\textsuperscript{a),} Mingjiang Wang\textsuperscript{b),} and Ming Liu

Harbin Institute of Technology, Shenzhen, HIT Campus, University Town of Shenzhen, Shenzhen, Guangdong, China
\textsuperscript{a)} zhaoboya@stu.hit.edu.cn
\textsuperscript{b)} mjwang@hit.edu.cn, Corresponding Author

Abstract: In this paper, we propose a CGSA (Coarse Grained Spatial Architecture) which processes different kinds of convolution with high performance and low energy consumption. The architecture’s 16 coarse grained parallel processing units achieve a peak 152 GOPS running at 500 MHz by exploiting local data reuse of image data, feature map data and filter weights. It achieves 99 frames/s on the convolutional layers of the AlexNet benchmark, consuming 264 mW working at 500 MHz and 1 V. We evaluated the architecture by comparing some recent CNN’s accelerators. The evaluation result shows that the proposed architecture achieves 3× energy efficiency and 3.5× area efficiency than existing work of the similar architecture and technology proposed by Chen.

Keywords: convolutional neural network, accelerator, AlexNet

Classification: Integrated circuits

References

1 Introduction

Machine learning is a fundamental technology for recognition, detection and speech understanding, natural language processing applications. Specifically deep convolutional neural networks (CNNs), can achieve unprecedented accuracy for tasks as object recognition [1, 2, 3, 4], detection [5, 6] and scene understanding [7]. These state-of-the-art CNNs requires up to hundreds of megabytes for filter weight storage and over 600k operations per input pixel. Now, a novel and powerful gaming AI use CNN [8] to understand the situation. In some commercial systems, CNN is utilized for improving the quality of the speech recognition and other services.

The large size of CNN poses both throughput and operation performance challenges to the processing hardware. Convolutions account for over 90% of the CNN operations [9]. The most typical approach to accelerate CNN is to use GPU [10]. It can process a matrix multiplication at very high speed. In addition to GPUs, FPGAs [11, 12, 13] and specific LSIs [14, 15, 16] have been proposed. By utilizing a specialized hardware structure for CNN, it can achieve higher throughput and operation performance, compared to GPU based approaches. In order to achieve high operation-performance CNN processing without compromising throughput, we need to design a new architecture and develop dataflows that support parallel processing with reducing the fetching data from off-chip DRAMs.

In this paper, we employ a Coarse Grained Spatial Architecture, which has an array of 16 reconfigurable processing elements (PES), and each PE support 9 MAC operations in parallel mode. Input data and filter weight parameter are read from external memory to internal SRAMs, and all the computing data are delivered from on-chip temporary SRAMs to each PE. The memory bandwidth pressure to the external memory is significantly reduced. The dynamic reconfiguration data of
CGRA for different sizes of convolutions is assigned by main controller block. In addition, by disabling unused PEs and SRAM blocks, the energy efficiency of this architecture is improved. The main advantage of our system is that it is Coarse Grained and that every PE is structured architecture. The architecture can implement with high efficiency of area, speed and low power.

We compare the architecture and dataflow in AlexNet CONV Layers with existing designs in (1) PE constitution, (2) register allocation, (3) energy consumption and (4) DRAM accesses. The characteristics of this work are:

(1). Allocate 9 multipliers in each PE for the first time to build structured PE units so that the area is compact.

(2). Using register array instead of scratch pad in PE units reduces data redundancy and the corresponding power consumption.

(3). Data SRAM is divided into parts and the unused parts in each stage are powered off to reduce energy consumption.

(4). Data are accessed as much as possible from SRAM so that DRAM data access is limited.

We firstly highlight the need for CNN acceleration (Section 2), and present a overall architecture of the accelerator (Section 3.1). We then show the architecture of PE and CGSA array (Section 3.2) and storage structure of on-chip SRAMs (Section 3.3). We discuss our experiment results in Section 4 and subsequently conclude the paper with Section 5.

2 CNN background

2.1 The basics

A convolutional neural network (CNN) is constructed by multiple computation layers. Through the computation of each layer, a higher-level abstraction of the input data, called a feature map (fmap), is extracted to preserve essential and unique information. Modern CNNs are able to achieve superior performance by employing a very deep hierarchy of layers.

The primary computation of CNN is in the convolutional (CONV) layers, which perform high-dimensional convolutions. Several hundred CONV layers are commonly used in recent CNN models [4]. A CONV layer applies filters on the input fmaps (ifmaps) to extract embedded visual characteristics and generate the output fmaps (ofmaps) [17], which are shown in Fig. 1.

![Feature visualization of convolutional net trained on ImageNet](image)

**Fig. 1.** Feature visualization of convolutional net trained on ImageNet

For one frame of image, ifmaps in 3D are processed by a group of 3D filters in a CONV layer: each filter is a 3D structure consisting of multiple 2D planes, i.e., channels. In addition, there is a 1D bias that is added to the filtering results. The computation of a CONV layer is defined as
\[ O[n][x][y] = \text{ReLU} \left( B[n] + \sum_{k=0}^{C-1} \sum_{i=0}^{F-1} \sum_{j=0}^{F-1} I[k][Sx+i][Sy+j] \times W[n][k][i][j] \right), \]

\[ 0 \leq n < N, 0 \leq x, y < E, E = (H - F + S)/S \]

N is the number of 3D filters, also the number of ofmap channels; C is the number of ifmap or filter channels; H is the ifmap plane width or height; F is the filter plane width or height; E is the ofmap plane width or height. O, I, W and B are the matrices of the ofmaps, ifmaps, filters and biases, respectively. S is a given stride size.

The shape configurations of each layer in the AlexNet [1] are shown in Table I. The channel numbers of Layer 2, 4 and 5 are expressed with \( \times 2 \) which means each half of the input data is convoluted with one half of the filters.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input Data size</th>
<th>Padding</th>
<th>Channel num.</th>
<th>Filter size</th>
<th>Stride</th>
<th>Filter num.</th>
<th>Output Feature size</th>
<th>Max Pooled size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>227 x 227</td>
<td>0</td>
<td>3</td>
<td>11 x 11</td>
<td>4</td>
<td>96</td>
<td>55 x 55</td>
<td>27 x 27</td>
</tr>
<tr>
<td>Conv2</td>
<td>27 x 27</td>
<td>2</td>
<td>48 x 2</td>
<td>5 x 5</td>
<td>1</td>
<td>256</td>
<td>27 x 27</td>
<td>13 x 13</td>
</tr>
<tr>
<td>Conv3</td>
<td>13 x 13</td>
<td>1</td>
<td>256</td>
<td>3 x 3</td>
<td>1</td>
<td>384</td>
<td>13 x 13</td>
<td>/</td>
</tr>
<tr>
<td>Conv4</td>
<td>13 x 13</td>
<td>1</td>
<td>192 x 2</td>
<td>3 x 3</td>
<td>1</td>
<td>384</td>
<td>13 x 13</td>
<td>/</td>
</tr>
<tr>
<td>Conv5</td>
<td>13 x 13</td>
<td>1</td>
<td>192 x 2</td>
<td>3 x 3</td>
<td>1</td>
<td>256</td>
<td>13 x 13</td>
<td>6 x 6</td>
</tr>
</tbody>
</table>

A few fully-connected (FC) layers are normally stacked behind the CONV layers for classification purposes. Additional layers can be added optionally between CONV and FC layers, such as the pooling and normalization layers. Each of the CONV and FC layers is also immediately followed by an activation layer, such as a rectified linear unit (ReLU).

### 2.2 Challenges in CNN processing

In most of the commonly used CNNs, such as AlexNet [2] and VGG16 [3], CONV layers account for over 90% of the overall operations and generate a large amount of data movement. Therefore, they have a significant impact on the computational complexity, throughput and energy cost of CNNs.

Processing of the CONV layers poses two challenges: high efficient operation and data handling. The detail of each is described below.

1. **High efficient operation:** For AlexNet, the shape parameters shown in Table I means that the hardware architecture need be capable to process three different shapes at least, i.e. convolution in 3 x 3, 5 x 5 and 11 x 11. The accelerator must have strong computing power, adaptation to different kinds of convolutions by configuration, efficient hardware employment, reduced area and low power.

2. **Data Handling:** Reading inputs directly from DRAM for all PEs requires high bandwidth and causes high energy consumption. This issue can be alleviated by using different types of input data reuse: convolutional reuse, filter reuse and ifmap reuse, partial sum (psum) reuse.
3 The proposed architecture

3.1 Spatial architecture

Fig. 2 illustrates the architecture block diagram of the proposed spatial architecture in this paper for CNN processing. It consists of an accelerator and off-chip DRAM. The accelerator is mainly composed of a PE array, general-purpose registers (GPR), a data SRAM, a parameter SRAM, an accumulation array (ACC array) and a Psum SRAM. The image data and CNN parameters are firstly transferred from external DRAM to on-chip Data SRAM and Param SRAM, and then processed by the accelerator. The values of partial sum of 2D convolutions are stored in on-chip Psum SRAM, and the calculation results of each layer are written back to DRAM.

By configuration of PEs, the PE array support different kinds of convolution processing, e.g. $11 \times 11$, $5 \times 5$ and $3 \times 3$. The data SRAM is 154 KB, which is for input data and input feature maps of each layer. The Param SRAM is for convolutional parameters, including filter weights and bias values. The GPR stores data and parameters to be processed in the next clock cycle by the PE array. The ACC array consists of sixteen 32-bit adders, performing accumulation of partial sum and bias. The intermediate accumulation results are stored in Psum SRAM, and the results of each layer are transferred to DRAM.

The input image is 8 bit per pixel; the filter weight parameters and the convolution result of each layer are 16-bit fixed-point; the convolution partial sum delivered from PE array to the accumulator is 32-bit fixed-point. The data SRAM, Param SRAM and the Psum SRAM can be used to exploit data reuse. The input data and parameters of each layer are read only once from DRAM. This improves the data accessing energy.

3.2 PE calculation array

The PE Array consists of 16 PEs, while each PE supports 9 MAC operations in parallel. For each MAC, the multiplier and multiplicand are 16-bit. Each PE reads 9 data and 9 filter weights from GPR, and calculates and outputs the 32-bit sum result of 9 dot products of data and weights.
The PE is implemented by booth-coding, wallace tree and CPA (Carry Propagate Adder) as shown in Fig. 3. For a $3 \times 3$ convolutional operation, each PE contains data and weights corresponding to a window of 3 rows by 3 columns, producing one partial sum. For a $5 \times 5$ or $11 \times 11$ convolution operation, each PE contains data and weights of up to 9 points from the input feature maps and weight parameters, producing a part of one partial sum. The intermediate values of 31-bit carry and 31-bit sum are outputted for later use in the calculation of $5 \times 5$ convolution and $11 \times 11$ convolution.

Fig. 3. The architecture of the processing engine

Fig. 4 shows the architecture of PE array. It has 144 input data and 144 weights. Each group of 3 PEs produces the result of a $5 \times 5$ convolution by adding the outputs of these PEs together. In addition, the result of an $11 \times 11$ convolution comes from the sum of all $5 \times 5$ convolutions. The Psum bus is multiplexed from three buses, psum-3, psum-5 and psum-11, which stands for the corresponding convolution results.

Fig. 4. The architecture of the PE array
For a $3 \times 3$ convolution operation (AlexNet Layers 3, 4, 5), the sum result outputs of 16 PEs are connected to the psum-3 bus, producing 16 32-bit partial sum values.

For a $5 \times 5$ convolution operation (AlexNet Layer 2), the sum results by adding the intermediate values of carry and sum of each group of 3 PEs are connected to the psum-5 bus, producing five 32-bit partial sum values.

For an $11 \times 11$ convolution operation (AlexNet Layer 1), the sum result of all $5 \times 5$ convolution results is connected to the psum-11 bus, producing one 32-bit value.

### 3.3 The architecture of data SRAM

The data SRAM reuses input image data and feature maps, reducing DRAM accesses. Nevertheless, for different layers of CNN convolution operations, the required amount of image data and feature maps are different at every clock. For AlexNet CNN, the $11 \times 11$ convolutions of Layer 1 with stride $= 4$ need a image data throughput of $11 \times 4 \times 8\text{ bit} = 352\text{ bits}$, and the $5 \times 5$ convolutions of Layer 2 with stride $= 1$ require a throughput of $(5 + 5 - 1) \times 1 \times 16\text{ bit} = 144\text{ bits}$. The $3 \times 3$ convolutions of Layer 3, 4 and 5 with stride $= 1$ require a throughput of $(3 + 16 - 1) \times 1 \times 16\text{ bit} = 288\text{ bits}$.

Therefore, to meet the requirements above, a structure of Data SRAM is proposed, shown in Fig. 5. The Data SRAM is composed of 11 independent 32-bit SRAM with length of 3753, S0, S1 to S10. Data SRAM support data bandwidth of $32 \times N\text{ bit}$ ($1 \leq N \leq 11$) according to configurations.

For AlexNet Layer 1, the input image data has 3 channels, with $227 \times 227$ size. Fig. 6a shows that 3 channels of input data of AlexNet Layer 1 are expanded and divided into groups of 11 columns, which corresponds to SRAM slices shown below. Different columns of image data in the same group are stored in different SRAMs in order to access in parallel. Each element of the Data SRAM stores 4 neighbouring pixels in one column, i.e. 32 bits.

For AlexNet Layer 2, the input feature maps with $27 \times 27$ size has 96 channels. One convolution arithmetic is implemented by the data from first half or second half 48 channels. We firstly expand 3D feature map data $(27 \times 27 \times 48)$ to 2D feature map data $(27 \times (27 \times 48))$. Fig. 6b shows the expanded 48 channels of feature map data in Layer 2, and the corresponding storage structure is shown below. Each element of the SRAM stores 2 neighbouring points in one row, also 32 bits. Only 5 SRAMs are necessary and enabled, so that power consumption is reduced.

For AlexNet Layer 3, the input feature maps with $13 \times 13$ size has 256 channels. We firstly expand 3D feature map data $(13 \times 13 \times 256)$ to 2D feature
map data \((13 \times (13 \times 256))\). Fig. 6c shows the expanded 256 channels of feature map data in Layer 3 and the corresponding storage structure. Each element of the SRAM stores 2 neighbouring points in one row, also 32 bits. Only 9 SRAMs are necessary and enabled, so that power consumption is reduced.

For AlexNet Layer 4 and 5, the input feature maps with \(13 \times 13\) size has 384 channels. One convolution arithmetic is implemented by the data from first half or second half 192 channels. We expand 3D feature map data \((13 \times (13 \times 256))\) to 2D feature map data \((13 \times (13 \times 192))\). Only 9 SRAMs are necessary and enabled.

### 3.4 Dataflow

The dataflow design of convolution operations for AlexNet are explained below, which is mainly involved in the process of data transmitting from SRAM to GPR. Because the dataflow of \(5 \times 5\) convolution and \(3 \times 3\) convolution are similar, only the former is shown.

For the weight sharing property of CNN, the weight values are read by channels of groups, while the corresponding feature values are read and then calculated. For
one group of weights, the convolution results of each channel are stored tempor-
arily as a partial sum in the Psum SRAM, to be accumulated with the next channel;
and the cumulative results of all channels are outputted as a channel of the output
feature maps to the off-chip DRAM.

### 3.4.1 Convolution 11 × 11

The input data of Layer 1 in AlexNet is an image with 3 channels of 227 × 227
pixels (including zero padding), while each pixel is represented by 8 bits. The
dataflow for one channel is illustrated below as an example.

Fig. 7 shows a visualization of data reading from Data SRAM. During each
clock cycle, 4 rows of 11 columns of pixels are read. The areas marked with (1),
(2), (3) and (4) represent the data of the image accessed during each clock cycle,
and the figure on the right shows the corresponding locations in the Data SRAM.
The PE array starts the 11 × 11 convolution operations after the third reading cycle,
which can be thought as an 11 × 11 convolution window.

![Fig. 7. The sliding window of the convolution process in Layer 1 and the corresponding locations in Data SRAM](image)

During the 57th reading cycle, the convolution window reaches the bottom of
the image for the first time. Then the window moves four pixels to the right and
begins to move up, and this process continues until all pixels in this channel are
read, as is shown in Fig. 7(b). The corresponding locations in the Data SRAM are
shown in the right figure.
For Parameter SRAM, the GPR reads the weight values in the first three clock cycles during reading each channel of data.

3.4.2 Convolution 5 \times 5

The PE array is able to perform five 5 \times 5 convolution operations for one time. Five adjacent windows in a line corresponds to 5 rows of 9 columns of pixels. Therefore, 18B (1 \times 9 \times 16 \text{bit}) data are read during each clock cycle from the Data SRAM.

The areas marked with (1), (2), (3) and (4) in Fig. 8(a) represent the data read during each clock cycle, of which the corresponding locations in the Data SRAM are shown in the right figure. The two rows on the top filled with zero are the padding of the convolution operation. Since the padding values is not stored in the SRAM, they are automatically processed in the GPR. The PE array begins the 5 \times 5 convolution operations after the third reading clock cycle.

![Fig. 8. The sliding window of the convolution process in Layer 2 and the corresponding locations in Data SRAM](image)

The convolution window reaches the bottom of the feature map matrix (including two rows of zero padding) during the 27th clock cycle. Then the window is moved back to the top of the matrix and shifted to the right to start moving down again, as shown in Fig. 8(b).

4 Experiments and discussion

The proposed design is synthesized in 65 nm TSMC CMOS. It achieves 500 MHz core clock frequency, implementing a frame rate of 99.2 fps on the AlexNet CONV layers, consuming 264 mW at 1 V. The specifications of this accelerator is shown in Table II.
Table II. Accelerator specifications

<table>
<thead>
<tr>
<th></th>
<th>TSMC 65 nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td></td>
</tr>
<tr>
<td>Core Area</td>
<td>4.0 mm²</td>
</tr>
<tr>
<td>On-chip SRAM</td>
<td>180 kB</td>
</tr>
<tr>
<td># of PE</td>
<td>16</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>152 GOPS</td>
</tr>
<tr>
<td>Power</td>
<td>264 mW</td>
</tr>
</tbody>
</table>

Table III shows the comparison with other ASIC implementations of CNN. Cavigelli [14] presented an accelerator implemented in a 65-nm CMOS technology, running at 12b at an average performance of 145 GOPS. The accelerator used a specific $7 \times 7$ convolutional engine. This paper uses a PE array that is capable to process $11 \times 11$, $5 \times 5$ and $3 \times 3$ convolutions, improving flexibility.

Table III. Comparison of this paper with previous published ConvNet implementations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>65 nm</td>
<td>65 nm LP</td>
<td>40 nm LP</td>
<td>65 nm</td>
</tr>
<tr>
<td>Gate Count [NAND-2]</td>
<td>912k</td>
<td>1852k</td>
<td>1600k</td>
<td>910k</td>
</tr>
<tr>
<td>Core Area</td>
<td>1.13 mm²</td>
<td>12.25 mm²</td>
<td>2.4 mm²</td>
<td>4.0 mm²</td>
</tr>
<tr>
<td>On-Chip SRAM</td>
<td>43 kB</td>
<td>181.5 kB</td>
<td>148 kB</td>
<td>180 kB</td>
</tr>
<tr>
<td># PE units</td>
<td>196</td>
<td>168</td>
<td>256</td>
<td>16</td>
</tr>
<tr>
<td># multipliers/PE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>1.2 V</td>
<td>1 V</td>
<td>1.1 V</td>
<td>1 V</td>
</tr>
<tr>
<td>Nominal Frequency</td>
<td>500 MHz</td>
<td>200 MHz</td>
<td>204 MHz</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>196 GOPS</td>
<td>67 GOPS</td>
<td>102 GOPS</td>
<td>152 GOPS</td>
</tr>
<tr>
<td>Average Performance</td>
<td>145 GOPS</td>
<td>60 GOPS</td>
<td>74 GOPS</td>
<td>141 GOPS</td>
</tr>
<tr>
<td>Word Bit-width</td>
<td>12 bits fixed</td>
<td>16 bits fixed</td>
<td>16 bits fixed</td>
<td>16 bits fixed</td>
</tr>
<tr>
<td>Power (AlexNet)</td>
<td>-</td>
<td>278 mW</td>
<td>76 mW</td>
<td>264 mW</td>
</tr>
<tr>
<td>Throughput (AlexNet)</td>
<td>-</td>
<td>34.7 fps (34.7 fps)(^{(1)})</td>
<td>47 fps (46.1 fps)(^{(1)})</td>
<td>99.2 fps (39.7 fps)(^{(1)})</td>
</tr>
<tr>
<td>Energy efficiency (AlexNet)</td>
<td>-</td>
<td>125 frame/J</td>
<td>625 frame/J</td>
<td>376 frame/J</td>
</tr>
<tr>
<td>Normalized Area</td>
<td>1.31 mm²</td>
<td>12.25 mm²</td>
<td>6.34 mm²(^{(2)})</td>
<td>4.00 mm²</td>
</tr>
<tr>
<td>Normalized Area Efficiency (AlexNet)</td>
<td>-</td>
<td>2.83 frame/s/mm²</td>
<td>7.27 frame/s/mm²</td>
<td>9.93 frame/s/mm²</td>
</tr>
<tr>
<td>Active of multipliers</td>
<td>-</td>
<td>88%</td>
<td>-</td>
<td>94%</td>
</tr>
<tr>
<td>Buffer Data Access</td>
<td>-</td>
<td>208.5 MB</td>
<td>-</td>
<td>335.6 MB</td>
</tr>
<tr>
<td>DRAM Data Access</td>
<td>-</td>
<td>15.4 MB</td>
<td>-</td>
<td>6.30 MB</td>
</tr>
</tbody>
</table>

(1) 200 MHz Normalized Throughput: $T_p^{200M} = T_p/(f_{clk}/200\text{ MHz})$
(2) 65 nm Normalized Area: $A_{65nm} = A_{40nm}/(40/65)^2$
Chen [15] presented an accelerator implemented in a 65-nm CMOS technology, running at 16b on the AlexNet benchmark at a throughput of 34.7 frames/s on the CONV layers. This work allocates 9 multipliers in each PE to build structured PE units so that the area is compact. Moreover, the Data SRAM is divided into parts and the unused parts in each stage are powered off to reduce energy consumption. Data are accessed as much as possible from SRAM so that DRAM data access and power is limited. Using register array instead of scratch pad in PE units reduces data redundancy and the corresponding power consumption.

Moons [16] proposed a 40-nm application-specific instruction set processor (ASIP) chip, running at programmable precision on the AlexNet benchmark at a throughput of 47 frames/s on the CONV layers. Due to different hardware platforms, it has better flexibility and energy efficiency than the work proposed in this paper. However, this paper allocates 9 multipliers in each PE to build structured PE units so that the area is compact and the area efficiency is improved.

The latency and power consumption together are considered for fair comparison in the perspective of energy efficiency. As a result, the implemented accelerator shows 376 frame/J over the AlexNet-benchmarked reference, which is 3× as high as the accelerator [15] shows.

The throughput and area together with normalized numbers (core frequency and technology) are considered for fair comparison in the perspective of area efficiency. This work achieves 9.93 frame/s/mm², which is 3.5× improvement over [15] and 36% advantage over [16] on the area efficiency.

5 Conclusion

This paper presents a coarse grained spatial architecture for CNN accelerators with high PE operation efficiency, maximizing input data reuse of filters and feature maps, while minimizing partial sum accumulation cost simultaneously. This design is synthesized in 65 nm TSMC, achieving 500 MHz core clock, and implements a frame rate of 99.2 fps on the AlexNet CONV layers.

Compared with existing work of similar architecture using AlexNet convolutional layers as a benchmark, the accelerator is 3× energy efficient and 3.5× area efficient on normalized area efficiency. The proposed architecture can achieve the fast processing rate, small output latency, small hardware, and low power consumption, simultaneously.

Acknowledgments

The research work was supported by the Infrastructure Research Program of Shenzhen (Grant No. JCYJ20130329153600746) and the Supporting Platform Program of Guangdong Province (Grant No. 2014B0909B001).