FBNA: A Fully Binarized Neural Network Accelerator

Peng Guo*†, Hong Ma*, Ruizhi Chen*†, Pin Li*, Shaolin Xie*, Donglin Wang*
* Institute of Automation, Chinese Academy of Sciences, Beijing, China
† School of Computer and Control Engineering, University of Chinese Academy of Sciences, China
Email: {guopeng2014, hong.ma, chenruizhi2014}@ia.ac.cn

***Abstract*—In recent researches, binarized neural network (BNN) has been proposed to address the massive computations and large memory footprint problem of the convolutional neural network (CNN). Several works have designed specific BNN accelerators and showed very promising results. Nevertheless, only part of the neural network is binarized in their architecture and the benefits of binary operations were not fully exploited. In this work, we propose the first fully binarized convolutional neural network accelerator (FBNA) architecture, in which all convolutional operations are binarized and unified, even including the first layer and padding. The fully unified architecture provides more resource, parallelism and scalability optimization opportunities. Compared with the state-of-the-art BNN accelerator, our evaluation results show 3.1x performance, 5.4x resource efficiency and 4.9x power efficiency on CIFAR-10.**

***Index Terms*—CNN, BNN, FPGA, Accelerator**

I. Introduction

To facilitate the deployment of convolutional neural network (CNN), many quantization based approaches have been proposed to reduce the storage requirement of CNNs, and one extreme low-bit scheme of them is binarized neural networks (BNN). In 2016, Courbariaux and Bengio first put forward BNN which constrains both weights and activations to +1 and -1, which can theoretically compress the full precision CNN by 32x [1]. The initial BNN models achieves comparable accuracy on small datasets like MNIST and Cifar10, and recent researches [2], [7] have significantly improved the BNN's accuracy on large datasets such as ImageNet, which further makes BNN an attractive architecture for the embedded and mobile platforms which have limited resource budget.

Compared to CNN, the multi-bits multiplications are replaced with 1-bit XNOR operations in BNN, which saves large hardware resources and greatly reduces the power consumption. Furthermore, as the weights in BNN are only 1-bit, the on-chip memory and bandwidth requirements are also reduced dramatically, providing new micro-architecture optimization opportunities. Previous work has taken advantage of these features and designed dedicated BNN accelerators [3]–[6], [8], [9]. The result shows great resource-efficiency and power-efficiency.

However, the benefits of BNN were not fully exploited in their work. As the input feature maps (ifmaps) of BNN's first layer are commonly float number, previous work introduces specific hardware to handle the computation of the first layer, which brings unnecessary loss of throughput and resource efficiency. Besides, the convolutional (Conv) layers are normally padded with 0 during the forward pass. To maintain the accuracy, they need two bits to represent ± 1 and 0, which requires more resource overhead.

Our work addresses the aforementioned problems. We first present a scheme to binarize the inputs of the first layer. Moreover, with negligible accuracy loss, we propose an odd-even padding scheme to represent the padding bits with $+1$ and -1. With these algorithm optimizations, we propose FBNA, which is the first fully-binaried CNN accelerator based on our knowledge. Compared with CNN, the binary operations of BNN are significantly simplified, which brings the possibility of a greater degree of parallelism. We present a unified Shuffle-Compute structure to take charge of the xnor operations of Conv layers. It can provide sustained high throughput for different ifmap sizes. Besides, it can be reused for the computations of fully-connected (FC) layers. At last, we implement FBNA on Xilinx Zynq 702 board and demonstrate its high energy efficiency and resource efficiency.

II. Hardware-oriented algorithm Optimization of BNN

A. *Binarize The First Layer*

Compared with CNN, a prominent advantage of BNN is that the most calculations can be done with bit-level operators. By representing $+1$ with a set bit and -1 with an unset bit, the multiplications of weights and ifmaps can be executed with xnor gates. However, the input data of the first layer is normally fixed-point number and most previous work adopts exclusive hardware design to tackle with this layer. For example, both [9] and [5] employ an independent fixed-point module for the first layer. [6] designs a shareable structure which can be reused as 32 1-bit multipliers or one full-precision multiplier. However, extra hardware will introduce higher design complexity and power consumption. Moreover, the whole performance is impaired by the first layer's calculations from two aspects. First, fixed-point multiplier consumes much more energy and resources than the xnor gate. So the number of the fixed-point multipliers is limited in most BNN hardware design and the calculations of the first layer cannot be very efficient. Second, the input layer normally has fewer channels, which is often one order of magnitude less than other Conv layers. For the accelerator designs with intra-outputs

parallelism on ifmap channels, the calculations of the first layer is hard to make full use of the hardware resources.

Binarization To address the problems brought by the first layer, we propose a two-step optimization scheme that consists of binarization and pruning. In binarization, the input fixed-point data of most networks in computer vision is normalized RGB value, which can be transformed to integer value between -128 and 128 and expanded to the sum of 256 ± 1 divided by 2. Let $\boldsymbol{A}$ be the ifmaps of the first layer, $\boldsymbol{W}$ be the weight matrix. Then the convolution of the first layer can be binarized as

$$\boldsymbol{A} \otimes \boldsymbol{W} = \boldsymbol{\alpha} * \sum_{i} \boldsymbol{B_i} \otimes \boldsymbol{W}, \tag{1}$$

where $\boldsymbol{B_i}$ is a binarized matrix and $\boldsymbol{\alpha}$ is a constant. A simplified example is illustrated in Figure 1(a,b,c).

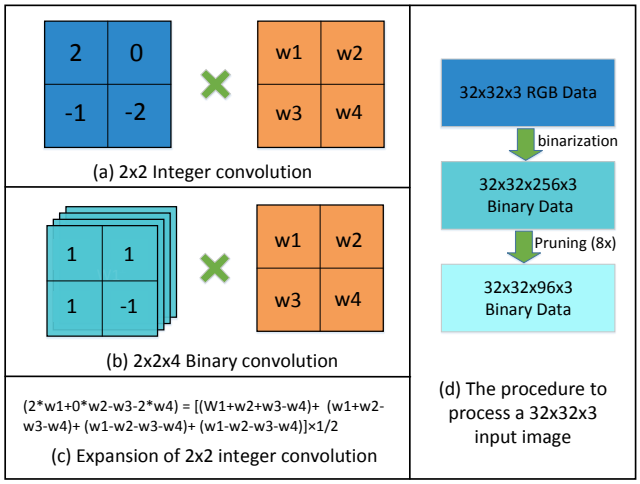

Figure 1. (a),(b),(c) illustrate the binarization of one 2x2 convolution with 2-bit data. (d) illustrate the full binarization-prunning procedure for one 32x32 RGB input.

Pruning With this binarization scheme, all Conv layers are binarized with large hardware resource saving. However, the channel number and xnor operations of the first layer increase exponentially. To resolve this problem, we prune away the low-bits before binarization. By pruning the low N bits of the input fixed-point data, the number of the expanded binarized channels would drop by 2^N times. We evaluate our method on BNN-Cifar10 and BNN-SVHN[1]. As shown in Figure 2, the error rate of BNN-SVHN increases from 2.97% to 3.14% when the low 4 bits are pruned away. It is interesting that the error rate of BNN-Cifar10 even decreases from 11.42% to 11.39% when we prune away the low 3 bits. A whole two-step optimization scheme is illustrated in Figure 1(d).

B. Odd-Even Padding

To preserve the accuracy, the ifmaps should be padded with zeros during convolution. Two bits are required to represent +1, -1 and 0, which makes the multiplication with one xnor gate impossible. Previous work [8], [9] pads the ifmaps directly with all +1, but as shown in Table 2, the error rate of BNN-Cifar10 and BNN-SVHN increases significantly.

[1]Both public at https://github.com/MatthieuCourbariaux/BinaryNet, but we reduce the neurons of BNN-SVHN's FC layers into 512 and retrain it.

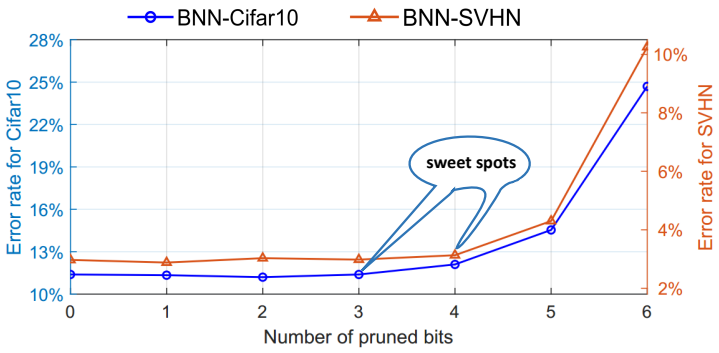

Figure 2. The accuracy of BNN-Cifar10 and BNN-SVHN when we prune away the low bits.

To neutralize the errors introduced by padding $+1$, we first propose the Odd-Padding and Even-padding scheme, which pads the ifmaps with interleaved $(+1,-1)$ or $(-1,+1)$ respectively. As shown in Table 2, the accuracy of Odd-padding significantly outperforms that of $+1$ padding. Then we propose the **Odd-Even Padding** scheme. As Figure 3 shows, the Odd-Even padding interleaves Odd-padding and Even-Padding for different ifmap channels, which can further neutralize the padding bits. In particular, because the RGB input channels share the same weight matrix, this scheme can get the same results as 0 padding on the binarized first layer. As shown in Table 2, The Odd-Even padding has almost the same accuracy as 0-padding.

TABLE I. Error rate of different padding methods

BNN Model	0-padding	All +1	Odd-padding	Odd-Even
BNN-SVHN	3.14%	3.36%	3.28%	3.15%
BNN-Cifar10	11.39%	13.23%	12.42%	11.25%

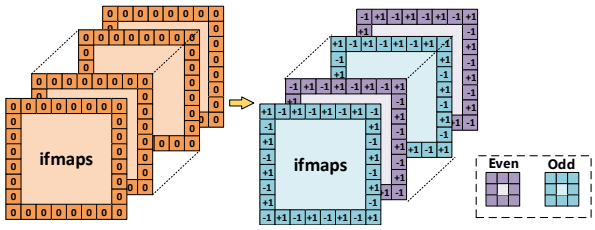

Figure 3. Even padding and Odd padding is interleaved to replace the zero padding.

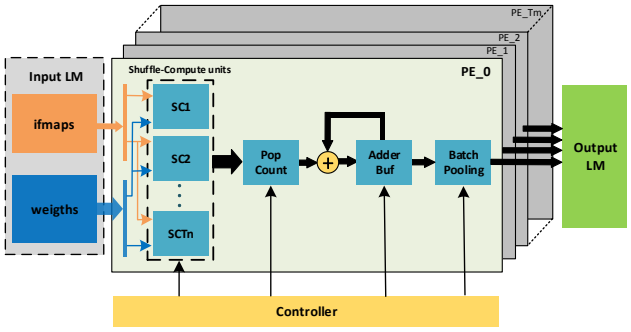

Figure 4. Architectural block diagram of FBNA.

III. Accelerator Architecture

A. *overview of FBNA*

The block diagram of FBNA is illustrated in Figure 4, which is composed of Tm processing elements, the controller, the input local memory (LM) and the output LM. During the computation of each layer, different PEs load same ifmaps from the input LM, generate independent output channels for Conv layer or respective vector segmentation for FC layer, and then write the intermediate results back to the output LM.

Each PE consists of four modules: Tn Shuffle-Compute units (SC), Popcount, Adder Buffer and BBP (BatchNorm, Binarize, and Pooling) unit. For BNNs, the primary computations of Conv layers and FC layers are convolutions and dot products respectively, which can be both decomposed into xnor operations and additions. During the calculation, SC units are responsible for the xnor operations while Popcount module is responsible for the subsequent additions. There are Tn SC units in one PE and each of them processes different ifmap channels concurrently. Details of SC unit will be explained in the next subsection. The popcount module sums the binary result of Tn SC units by counting the number of value 1, then the partial sum is further accumulated in the Adder Buffer. The batch norm, binarize and pooling operations are performed subsequently in the BBP module. Two techniques introduced in [8] are adopted to optimize BBP. The first combines the batch norm and binarize operation into direct comparison. The second swaps the pooling operation and the direct comparison, which transforms the integer comparison into a boolean OR.

B. *Shuffle-Compute Unit*

SC unit is the most critical component of the PE which is responsible for the multiplications of Conv layers and FC layers. As shown in Figure5 (a), SC includes shuffle unit, ifmaps buffer, weight buffer and parallel binary convolver (PBC). Both ifmaps buffer and weight register are connected to the local memory with a Ni bit bus. The PBC module takes a 3xNi array structure and each position of the array corresponds to one xnor gate.

During the computation of Conv layers, each 3x3 block of PBC corresponds to the operands of one 3x3 convolution and adjacent blocks share 6 ifmap pixels, thus the multiplication of Ni convolutions can be accomplished per cycle. Although the ifmaps for each cycle need 3xNi ifmaps, PBC only need to load Ni ifmaps from input LM per cycle. For example, when Ni is 64 and the size of ifmap is 32x32, in the first cycle the ifmaps' first three lines are required to calculate ofmaps' first two lines, then the 2nd to 5th lines are needed to calculate the third and fourth lines in next cycle. But the SC unit only need to load the 4th and 5th lines of ifmaps with the help of ifmaps buffer and shuffle unit.

For the computation of FC layers, the ifmap buffer and the weight buffer both take Ni pixels from local memory and the dot product of Ni data can be calculated in PBC per cycle.

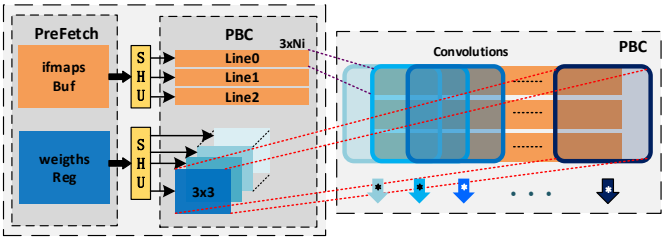

Figure 5. Block diagram of Shuffle-Compute Unit - (a) overall architecture of SCs; (b) block diagram of PBC with 3 rows and Ni columns.

IV. Experiment

A. *Experimental Setup*

To evaluate our design, we implement FBNA on Xilinx Zynq ZC702 evaluation board. The chip on this board integrates a Xilinx Artix-7 FPGA and a dual-core ARM Cortex-A9 MPCore. FBNA is implemented with verilog and mapped to the PL part. We evaluate different configurations of FBNA and finally take Ni = 64, Tm =1 and Tn = 16 in the following experiments. All synthesis results are obtained from Xilinx Vivado 2016.4. We measure the chip power through the PMBus.

The fully binarized BNN-Cifar10 and BNN-SVHN which both have 6 Conv layers and 3 FC layers are taken as our experimental networks. The BNN-Cifar10 has a total of 1386 GOP computation and 1.7 MB weights, which can achieve 88.61% accuracy on Cifar10. The BNN-SVHN is a simplified version of BNN-Cifar10. It has only 346 GOP and 0.45 MB weights, achieving 96.9% accuracy on SVHN dataset.

B. *Comparison*

As shown in Table 4, we first compare FBNA against two general platforms: the Intel 6700K CPU and the NVIDIA GTX1070 GPU. Then, we compare our design with two FPGA based BNN accelerators: FINN [8] and the accelerator proposed by Zhao et al. [9]. We use GOPS per kLUT and GOPS per watt to represent the resource efficiency and power efficiency.

CPU/GPU Both CPU and GPU are evaluated with theano on PC. The power of CPU is from Intel's datasheet and the power of GPU is reported with the $nvidia - smi$ command.

TABLE II. Comparison with other BNN platforms for SVHN and Cifar10

platform	Device	Dataset	Bit-width	Accu.	kLUTs	BRAM	Time per image (ms)	GOPS	Power (W)	GOPS/ kLUT	GOPS/W
CPU	intel 6700k	SVHN	-	97.1%	-	-	5.0	69	91	-	0.76
GPU	Nvidia GTX1070	SVHN	-	97.1%	-	-	0.13	2708	133	-	9.25
FINN [8]	Xilinx ZC706	SVHN	1-bit	94.9%	46.2	186	0.05	2465	3.6	53.3	684.7
FBNA	Xilinx ZC702	SVHN	1-bit	96.9%	29.6	103	0.155	2236	3.2	75	699
CPU	intel 6700k	Cifar10	-	88.58%	-	-	10.3	135	91	-	1.48
GPU	Nvidia GTX1070	Cifar10	-	88.58%	-	-	0.44	3380	147	-	24.7
FINN [8]	Xilinx ZC706	SVHN	1-bit	80.0%	46.2	186	0.05	2465	3.6	53.3	684.7
zhao [9]	Xilinx ZC702	Cifar10	2-bit	88.54%	46.9	94	5.94	208	4.7	4.43	44.2
FBNA	Xilinx ZC702	Cifar10	1-bit	88.61%	29.6	103	1.92	722	3.3	24	219

Compared with CPU, our design achieves about 32x better performance and 920x lower power on SVHN while 5x better performance and 148x lower power on Cifar10. The large variance between the two models is due to the capacity of on-chip memory, in which the Cifar10 model exceeds the capacity and the performance is limited by the bandwidth. Although the performance of GPU outperforms our design, we achieve 75x and 8.9x power efficiency on SVHN and Cifar10 respectively.

FINN The results of FINN [8] are evaluated on a tiny network with only 0.19MB parameters. For the SVHN dataset, it achieves similar performance as ours, while our design only uses 64% LUTs and 55% BRAM, and achieves significantly better accuracy. For the Cifar10 dataset, although they achieve much better performance and resource efficiency, their accuracy drops to 80.0% while ours is 88.6%. The different comparison result for these two networks is mainly due to that FINN [8] take the same model for the two datasets and this model is small enough to be fully put on the chip.

Zhao et al. The design in [9] is implemented with the same low-cost FPGA platform and the same BNN-Cifar10 network model on Cifar10 as FBNA. Both designs are implemented under 143 MHz and show similar accuracy. As shown in Table 4, compared with 5.94 ms, FBNA only needs 1.92 ms per image, which is 3.1x speedup. Besides, FBNA achieves 5.4x resource efficiency and 4.9x power efficiency respectively.

The detailed comparison is illustrated in Figure 6. FBNA achieves 10x, 4.4x, and 1.8x speedup for the first Conv layer, other Conv layers, and FC layers respectively. The performance improvements can be attributed to the following points: First, the algorithm optimizations enable FBNA to be a fully binarize hardware while the reference design still needs 2-bit computations. Second, FBNA unifies the computation of all layers and proposes the high-efficient SC units, while the reference design embeds dedicated hardware for Conv 1 layer and FC layers. Third, the memory system of FBNA enables higher utilization of external memory bandwidth.

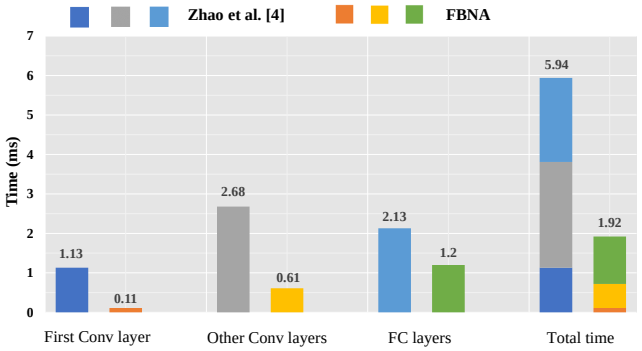

Figure 6. Detailed timing comparison with [9]

V. Conclusion

In this paper, we propose a fully binarized convolutional neural network accelerator (FBNA). With the hardware-oriented algorithm optimizations, we are the first to binarize all layers and construct a unified hardware design for BNNs. The SC units are proposed to support high parallelism on all layers. We implement our design with Xilinx ZC702 and fully evaluate its performance with two datasets. On Cifar10, it achieves 722 GOPS overall performance, 24 GPOS/KLUT resource efficiency, and 118 GOPS/watt power efficiency, which has 3.1x, 5.4x, and 4.9x improvements of the state-of-art works. Future work will focus on implementing our design with larger BNN models on larger dataset like imageNet.

References

[1] Courbariaux et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[2] Li et al. Performance guaranteed network acceleration via high-order residual quantization. *arXiv preprint arXiv:1708.08687*, 2017.

[3] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei. Fp-bnn: Binarized neural network on fpga. *Neurocomputing*, 275:1072–1086, 2018.

[4] D. J. Moss, E. Nurvitadhi, J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P. H. Leong. High performance binary neural networks on the xeon+ fpga platform. In *Field Programmable Logic and Applications (FPL), 2017 27th International Conference on*, pages 1–4. IEEE, 2017.

[5] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato. A lightweight yolov2: A binarized cnn with a parallel support vector regression for an fpga. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 31–40. ACM, 2018.

[6] Nurvitadhi et al. Accelerating binarized neural networks: comparison of fpga, cpu, gpu, and asic. In *Field-Programmable Technology (FPT), 2016 International Conference on*, pages 77–84. IEEE, 2016.

[7] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, pages 2625–2631, 2017.

[8] Umuroglu et al. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74. ACM, 2017.

[9] Zhao et al. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 15–24. ACM, 2017.