Implementation of Data-optimized FPGA-based Accelerator for Convolutional Neural Network

Mannhee Cho
School of Electronic and Electrical Engineering
Hongik University
Seoul, Korea
mhc9840@gmail.com

Youngmin Kim
School of Electronic and Electrical Engineering
Hongik University
Seoul, Korea
youngmin@hongik.ac.kr

***Abstract*— Convolutional Neural Networks (CNNs) are widely used for image recognition, and FPGAs are considered suitable platform for CNNs due to their low power consumption and reconfigurability. While CNNs are mostly trained using floating point data type for high inference accuracy, fixed point data type can be used to reduce data size and take advantage of computation efficiency on FPGAs without any accuracy loss. In this paper, we propose an accelerator design for LeNet-5 CNN architecture [1] for MNIST handwritten digit recognition. The accelerator is synthesized with Xilinx Vivado High-Level Synthesis (HLS) tool (v2017.2), targeting xczu9eg-ffvb1156-2-i FPGA board. The proposed accelerator focuses on reducing latency and memory usage, and the performance is compared with a conventional floating point design. Our proposed accelerator can achieve latency reduction up to 90% and memory usage reduction up to 40% without any accuracy loss, compared to the conventional design.**

Keywords—Convolutional Neural Network; FPGA; High-level Synthesis; Accelerator

I. INTRODUCTION

Convolutional Neural networks (CNNs) are widely used in various applications such as image classification, for their high classification accuracy. For their implementation FPGAs are considered as suitable platform for CNNs due to their low power consumption and reconfigurability. CNNs however require large computations and resource. To fit small hardware, CNN accelerators need to be optimized to reduce memory size, resource usage, and power consumption. Many researches have been conducted to implement more efficient high-performance CNN accelerators on FPGA platforms [2][3].

When designing hardware designs for FPGA platform, High-level Synthesis (HLS) can be used to efficiently implement hardware accelerators [4][5]. HLS allows the user to write and synthesize register-transfer level (RTL) designs in high level language (e.g. C/C++ language). Vivado High-Level Synthesis (HLS) tool supports usage of fixed point data types and provides synthesis directives that can automatically optimize the design, such as pipelining and unrolling of loops.

In this paper, we propose an efficient hardware CNN accelerator for LeNet-5 CNN [1], a CNN architecture for digit classification using MNIST handwritten digit dataset. Our main goal is to optimize the accelerator in terms of latency and memory usage, without any loss in classification accuracy. Our proposed accelerator achieves zero accuracy loss, 90% latency reduction, and 40% memory (e.g., BRAM) usage reduction, compared to the conventional design without optimizations.

II. ACCELERATOR DESIGN

The accelerator is synthesized with Vivado High-Level Synthesis tool (v2017.2), targeting xczu9eg-ffvb1156-2-i FPGA board. We use MNIST handwritten digit dataset for classification.

A. LeNet-5 Architecture

LeNet-5 architecture, as shown in Fig. 1, consists of three convolution layers (C1, C3, and C5), two subsampling layers (S2, S4), one fully connected layer (F6), and an output layer. The network uses 32-bit floating point data for input and parameters. It receives 32x32 input image from MNIST handwritten digit dataset for classification. Subsampling layers perform average pooling with 2x2 filter, and each layer uses *tanh* activation function, setting output range to [-1, 1].

B. Implementation

We used Vivado HLS (v2017.2) tool to synthesize the accelerator design, targeting xczu9eg-ffvb1156-2-i FPGA board. The accelerator reads in 32x32 input image, weights and biases for each layer, and then outputs the results after computation.

From our experiments on data bit width for LeNet-5 architecture, we found that the minimal bit width we can use to achieve zero accuracy loss is 20 bits, with 1 sign bit, 5 integer bits, and 14 decimal bits as shown in the top bitstream in Fig. 2. Further reducing the length of integer bits or decimal bits resulted in accuracy loss. While further reducing data bit width can result in better performance, we wanted to keep zero loss in accuracy for the accelerator.

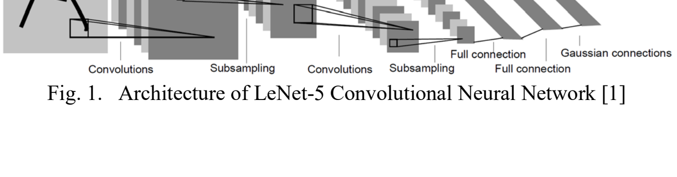

Fig. 1. Architecture of LeNet-5 Convolutional Neural Network [1]

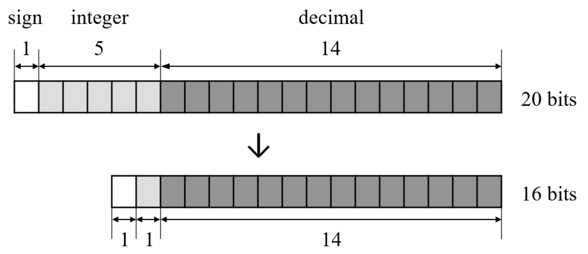

Fig. 2. Data bit width for accumulation (top) and transfer (bottom)

In addition, we further reduced the data bit width for storing and transferring data between layers. Since data transferred between each layer range from -1 to 1 due to *tanh* activation function after each layer, we can further reduce the length of integer bits from 5 bits to 1 bit, as shown in the bottom of Fig. 2. With a single sign bit and one integer bit, fixed point data can represent numbers -1, 0, and 1, along with the decimal values. As result, the accelerator uses 20 bit data for accumulation inside each layer, and 16 bit data for transfer between layers (i.e., input and output).

To optimize the algorithm, we used HLS pragmas (directives) provided in Vivado HLS tool. '#pragma HLS pipeline' and '#pragma HLS unroll' are used to parallelize the loops. Fig. 3 shows the algorithm of an optimized convolution layer with HLS pragmas applied.

Overall, our proposed accelerator design uses fixed point data type (20 bits for accumulation and 16 bits for transfer) and the algorithm is parallelized using HLS pragmas.

III. Experimental Results

The network is trained with MNIST handwritten digits, using floating point data. Trained parameters are encoded into 11-bit fixed point for inference test. We used 10,000 input images for testing, which are also encoded into 11-bit fixed point data. We compare the results of our proposed accelerator design (fixed point data with HLS pragmas) with the conventional design (floating point data without HLS pragmas).

Accuracy and performance comparison are shown in Table I and Table II, respectively. Total clock cycles and latency are estimated in Vivado HLS synthesis report to compare performance. As shown, our proposed design provides 90% latency reduction without any accuracy loss compared to the conventional design. Resource utilization is shown in Table III.

```
for (m = 0; m < depth_out; m++) {
for (n = 0; n < depth_in; n++) {
    for (row = 0; row < size_out; row++) {
    for (col = 0; col < size_out; col++) {
#pragma HLS pipeline
        for (row_k = 0; row_k < size_k; row_k++) {
#pragma HLS unroll
        for (col_k = 0; col_k < size_k; col_k++) {
#pragma HLS unroll
            o_buf [ ] = i_buf [ ] * k_buf [ ];
}}   }}   }}
```

Fig. 3. Pseudo code of an optimized convolutional layer

TABLE I. Comparison of Classification Accuracy

Implementation	Accuracy (correct/total)	Accuracy (%)
Conventional design	9863/10000	98.63
Proposed design	9864/10000	98.64

TABLE II. Comparison of Estimated Performance

Implementation	Clock cycles	Latency (ns)
Conventional design	3,895,572	32,800,716
Proposed design	410,758	3,581,810

TABLE III. Comparison of Estimated Resource Utilization in HLS

Resource	Conventional design		Proposed design	
	Used	Utilization	Used	Utilization
BRAM	163	8%	95	5%
DSP	79	3%	143	5%
FF	19195	3%	33585	6%
LUT	21260	7%	32589	11%

As shown, BRAM usage is reduced by 40% compared to the conventional design. Utilization of DSP, FF, and LUT are increased however, in order to achieve parallelism in the algorithm, increasing throughput and decreasing latency.

IV. Conclusions

Experiment results show that our proposed accelerator can achieve significant reduction in latency and memory usage compared to a conventional floating point data design. We can efficiently design a hardware CNN accelerator using an HLS tool in FPGA and optimized fixed point data type to further reduce the latency and memory size required for the accelerator.

Acknowledgment

This research was supported by the Basic Science Research Program, through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2017R1D1A1B03028065).

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, November 1998.

[2] C. Farabet, C. Poulet, J. Y. Han and Y. LeCun, "CNP: an FPGA-based processor for Convolutional Networks," 19th International Conference on Field Programmable Logic and Applications, pp. 32-37, August 2009.

[3] J. Qiu, et al., "Going deeper with embedded FPGA platform for Convolutional Neural Network," in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 26-35, Feburary 2016.

[4] C. Zhang, et al., "Optimizing FPGA-based accelerator design for Deep Convolutional Neural Networks," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 161-170, Feburary 2015.

[5] Y. Zhou and J. Jiang, "An FPGA-based accelerator implementation for Deep Convolutional Neural Networks," 2015 4th International Conference on Computer Science and Network Technology, pp. 829-832, December 2015.