Deep Learning Acceleration with a Look-Up-Table Based Memory Logic Conjugated System

Daisuke Ogawa*,**, Yoichi Sato***, Yasutaka Wada**, and Kanji Otsuka***

*Graduate School of Information Science, Meisei University, 2-1-1 Hodokubo, Hino, Tokyo 191-8506, Japan
**School of Information Science, Meisei University, 2-1-1 Hodokubo, Hino, Tokyo 191-8506, Japan
***Collaborative Research Center, Meisei University, 2-1-1 Hodokubo, Hino, Tokyo 191-8506, Japan

(Received August 23, 2018; accepted March 7, 2019, published June 19, 2019)

Abstract
This paper is the progressive study of previous papers presented at the IMPACT 2015 and ICEP 2018, and evaluates effectiveness and applicability of MLCS (Memory Logic Conjugated System) with a simple deep learning processing. NVIDIA, Google, Fujitsu Intel-Altera, Intel-Nervana and Renesas recently announced that 8 bits processing can keep efficient and flexible AI computation, peculiarly in deep learning. This paper discusses on the actual MLCS circuit implemented on a commercial FPGA for deep learning, and evaluate the circuit with perceptron method for deep learning. In the MLCS architecture, deep learning computations can be done as memory operations. Our architecture can achieve its high I/O bandwidth and low-power consumption with dynamic reconfiguration functionality, high-speed connection among logics and memory cells, and low implementation cost.

Keywords: LUT architecture, Deep learning, Seamless training and inference, High-performance, Low-power consumption, Memory logic conjugated system (MLCS)

1. Introduction
The Memory Logic Conjugated System (MLCS) has advantage in not only design and algorithmic flexibility, but also low-power and high-speed operation even in using conventional SRAM circuit thanks to dynamic reconfiguration ability.[1, 2] A defined area of memory cell can be utilized as LUT logic in some cases, and also, it can be used as memory in other cases depending on operation needs as shown in Fig. 1 (Dynamic reconfiguration). In this figure, green blocks represent memory cells used as LUT logic cells; blue colored blocks represent as cache memory. We can change their functionalities according to the required amount of computation logic and/or memory capacity. Also, logic cells and memory cells are adjacently to realize efficient communication between logic cells and memory cells.

Latest AI processors announced by NVIDIA,[3] Google,[4] Fujitsu,[5] Intel-Altera, Intel-Nervana[6] and Renesas[7] etc. are equipped with structured ALUs with low precisions due to the requirement for flexibility and simplicity for perceptron or CNN (Convolutional Neural Network) execution. For deep learning execution, most of perceptron neural networking procedure can be satisfied with simple 8 bit integer calculation.[5] Our MLCS is suitable for such a perceptron approach. Therefore, this paper evaluates the applicability and effectiveness of MLCS to deep learning operations. MLCS architecture is implemented in the FPGA to fit the simple deep learning operations for hand-written decimal letter recognition.

The basic network structure for the deep learning operation used in this paper is shown in Fig. 2(a), which has
single-layered structure of n input with the all-to-all connection between the layers. Each connection has its own weight. A signal receiving node (between Weight and Sum) gets a step of data from all previous layer node and accumulates (Sum, Transfer function) the data on the node as shown in Fig. 2(b). The result is just made for excited “1” or calm “0” state by the proper threshold in an instant. As over or under weightings prevents proper threshold, the adjustment is done by the updating weight and theta (Sum) in an instant on the stage of learning (Sum, Transfer function). After all weight parameters are randomly set, data adjustment is repeated to make the network provide the correct values. This iterative learning step could make proper weight values automatically. Some hand writing letters are introduced as the execution example, and they are examined in that huge number of cycles of the learning to achieve the right hand-writing letter recognition.

One typical deep learning system, based on Google TPU,[4] is shown Fig. 3. Pre-processed input data (i.e., all image pixels) from left side and weight fetch from upper side weight memory are input to matrix multiplier unit at first to multiply them, and then, the results are accumulated for activation. This operation flow is repeated for all structured cells by cycle with 700 MHz in the case of TPU. Though TPU hardware consists of relatively large matrix multiplier and large parallel accumulators, deep learning requirements vary from small to huge depending on the network structure and size. Therefore, communication structure and its bandwidth should be adjusted corresponding to user application requirements.

MLCS has dynamic reconfiguration functionality as mentioned above, and can realize an optimized configuration, subsequent power saving ability and quick response. In this paper, we introduce and evaluate the MLCS architecture to show its power-performance efficiency.

2. MLCS for Deep Learning

2.1 MLCS basic structure

The logic element of MLCS, which we call as MLCS basic cluster, is shown in Fig. 4. The MLCS basic cluster consists of functional memories as LUTs (Look Up Table), function-control registers, buses and controllers; the MLCS basic cluster is arranged in tile structure and connected directly with their neighbors via logic area bus. In our system, the buses are arranged with one global bus line, so the number of buses can be reduced by giving multiple function configuration to the MLCS basic cluster. For that reason, communication cost among neighbor blocks can be reduced. Thus, this tightly connected structure can reduce the number of buses, the space among clusters, and the total chip size. Also, the required bus length can be shortened by optimizing the cluster arrangement for each application. This optimized array structure is suitable for deep learning iterative operations.

![Fig. 2(a) Perceptron.](image)

![Fig. 2(b) Multilayer Perceptron.](image)

![Fig. 3 TPU Structure.](image)

![Fig. 4 Logic element of MLCS (MLCS basic cell).](image)
On the other hand, in FPGA shown in Fig. 5, there are logic blocks consisting of many small LUTs and local routing networks, among them with cross-bar type switching blocks. Therefore, the local routing networks contain a lot of buses and consume major area on FPGA. This prevents us from utilizing chips area effectively.

Another important characteristic of the cluster-based (integrated MLCS block) MLCS structure is its less complex design. Figure 6 shows the concept of our algorithmic consideration. Multiplication of 4-bits can execute one basic cluster of 256 words × 8 bits structure and adder function can operate two or four basic clusters depending on the carry bits. NAND, NOR, and others as four arithmetic operations also use the basic cell that is a fundamental logic unit with flexibility.

### 2.2 Applying MLCS for deep learning

We designed the three layer perceptron structures shown in Fig. 2(b). This structure is applied to do image recognition of a hand-writing decimal letter for the 8 × 8 divided pixels. The perceptron cells are constructed of 64 as the first layer, 20 cells for the second hidden layer, and 10 output cells for the third layer [denoted as “input”, “hidden”, and “output” in Fig. 2(b)]. Each cell in adjacent layers are connected in an all-to-all manner. The hidden layer and output layer accumulate all the cell data (pixels) in the first and second layers after weighting as shown in Fig. 7. Input data transfers data to the circuit for product-sum thru shift register. Then, the circuit executes and corrects data by previous weight values in weight memory. These execution cycles are repeated until the weight value reaches saturation. Finally, the saturated value becomes the output.

For the deep learning set of multiplier/accumulator for MLCS, one design of the circuit for 4 bits is as shown in Fig. 8 that is based on the 4-bit LUT units (see bright blue rectangle). Cycle times in decimals were required for the execution; therefore, that figure is fixed. Weight and cost values need to two and three decimals for the equation respectively. The accumulator has carryover space, a result of which is accumulation that causes 16 bits as shown in Fig. 9. However, output should be remade to 4 bits by the execution down of the digit according to the 4-bit system.

If we design 4-bit multiplier for MLCS and logical calculation, MLCS results in more contrast to make advantage as shown in Figs. 8 and 10. These figures show a comparison between LUT step calculation, and it is a conventional logical one. It can be seen as a slightly larger chip area of
MLCS than the logic circuit needed for area of SRAM memory if we make FPGA chip; therefore, both chip areas are becoming equivalent with the consideration of bus area.

In general, pixel composition of pictures is larger than 64 pixels, repeated executions should be done as both MLCS circuits and logical calculation circuits. In this case, another accumulator and operation latency are needed to depend on the repeated cycles with hierarchy structure. That needs dynamic change in shifting each layer, and should be built for such complicated execution jobs resulting in higher power consumption. Thus, LUT architecture of the MLCS has more advantage with the low-power consumption than any other system.

3. Performance Evaluation of FPGA Based MLCS Implementation

3.1 MLCS implementation for MAX® 10 FPGA board

In this study, we employ a commercial FPGA evaluation board -DK-DEV-10M50-A MAX® 10 Intel/Altera- to implement LUT based MLCS with 1-port SRAMs and 4 bit parallel adders and multiplication logic.

Table 1 summarizes the three implementations of product-sum methods, which are configured into the FPGA, used of the evaluation in this paper. There are three types of product-sum methods: 1. Logical Calculation, 2. Design Tool (macro-model design), and 3. LUT base design for MLCS (MLCS). The FPGA design tool method is not examined in this study because of it is similar as the logical calculation. Certain SRAM blocks are used to realize these circuits.

The implementation of MLCS in Max® 10 is shown in Figs. 11 and 12, comparing with that of logical calculation. The major difference between the MLCS and the logical calculation is the location of the LUT embedded in the FPGA. The LUT for the MLCS is implemented in the SRAM block area (memory blocks). On the other hand, the LUT for the logical calculation is inside of the logic circuit (logic array). The MLCS in the SRAM consists of memory blocks for use as a work memory and cache.

Table 1 Three types of product-sum method.

| Product-sum method | programming | Characteristics | Number of logic element [LEs] | Memory area in FPGA [Memory Bits] | Clock speed for perceptron [MHz] |
|--------------------|-------------|-----------------|-----------------------------|----------------------------------|-------------------------------|
| Logical Calculation| Verilog     | Logic element design | 3,035                       | 160                              | 160                           |
| FPGA Library       | VHDL        | Design Tool     | 3,060                       | 150                              | 150                           |
| LUT by MLCS        | Verilog     | Communication in logic element. 350 MHz/7 cycles | 2,536                       | 96,256                           | 330                           |

*1 Number of logic element [LEs]
*2 Memory area in FPGA [Memory Bits]
*3 Clock speed for perceptron [MHz]
memory, although the SRAM is generally implemented in the area where the LUT is not implemented.

We need additional logic for the serial interface between FPGA and host PC side. This interface logic between the FPGA board and the host PC occupies some circuit area and raises communication cost while executing the deep learning operations. The data which are obtained from the sum-product execution in the FPGA is transferred to the host PC side by serial communication and control units, then the cost function values are displayed on the PC monitor by VBA on the PC side as shown in Fig. 13.

In terms of the number of consumed logic elements, there is no significant difference between the MLCS and logical calculation. The LUT for the MLCS is implemented in the memory blocks. Therefore, the MLCS gives us higher frequency than the logical calculation as shown in Table 1.

Figure 14 shows the cost function value which is defined as

$$C_t = \frac{1}{2} \left( (t_1 - a_1^0)^2 + (t_2 - a_2^0)^2 + (t_3 - a_3^0)^2 \right)$$

“$t_1 - t_2$": Correct = 0, Other = 1, “$a_1^0 - a_1^0$": learning data

Here, $C_t$ means the error rate. This can be used to evaluate error rate difference between output data and reference data at each epoch during the learning process for each category of hand-written letters.

A smaller $C_t$ value increases the learning propriety, and a high value decrease the learning propriety. Also, if it learns similar letter feature, it is difficult to distinguish the letter feature, so the learning rate tends to be higher $C_t$. In this case, the learning propriety will not improve unless the number of learning iterating cycles is increased.

3.2 Evaluation results of MLCS for deep learning

Figures 14 and 15 show results of the MLCS’s learning profiles for the cost value of each hand-written letter and their accumulation value of $C_t$. Here, logical calculation execution profiles are obtained as the same (not shown later because it is a duplication of another), because of the same learning profile due to the same algorithmic approaches. Fig. 14 shows typical results on partially hand-written letters of 10 letters in 100 patterns. To recognize every letter shape, large number of learning iteration cycles are executed under the condition that the number of patterns is 100 and epoch is 1,000.

Each learning process takes more than 70 hours to complete. Figure 16 shows that MLCS achieves eight hours reduction in total execution times compared to logical calculation. Though the learning operations are executed on the FPGA side, controlling tasks -which take almost all the time from execution and communications- are executed on the host PC and taken cycles at the same on MLCS and logical calculation methods.
The learning/inference results are shown in Table 2, which indicates the same result between logical calculation and MLCS because initial setting values of each weight are the same when we use the same training data set. In this evaluation, propriety rate of the trained iteration cycles is from 0.95 to 0.7 depending on the letter to be recognized.

In our measurement, MAX® 10 power consumption of measurement is only 2.2 W on average for total running state. Simple deep learning models can be achieved through such lower power in either MLCS or logical calculation, and this makes the deep learning of FPGAs more attractive.

### 4. Conclusions

This paper evaluated the performance and applicability of MLCS architecture with deep learning operations. Through the performance evaluation with deep learning computations, this paper showed and discussed that MLCS can achieve a more flexible structure and higher performance in comparison with conventional serial multiplication based circuit. MLCS provides us with dynamic reconfiguration functionalities, and this makes it possible to maximize memory bandwidth while reducing wiring complexity, as one of the most challenging problems. With our MLCS architecture, we can easily solve the problems of memory bandwidth bottleneck and higher wiring power consumption. MLCS architecture is suitable for deep learning operation with its low power consumption, dynamic reconfigurable ability, simple memory matrix structure, and high speed operation. In the future, we would like to evaluate MLCS architecture with other applications, and to design and implement MLCS on ASIC.

### Acknowledgments

The authors would like to thank Mrs. Tetsuro Matsumoto (NEIDA), Shigeru Yamamoto (NSW), Takeshi Koyama (NSW), Takanori Nomura (NSW), Koshi Sato (Hikari Science), Kenya Yanagihara (Onoduka Co., Ltd.) and Ms. Machiko Yamamoto (Meisei Univ.) for their discussions to improve MLCS architecture and its implementation.

### References

[1] K. Otsuka and Y. Sato, “High Speed, Flexible, Robust and Low Power Processing Approach,” Proc. IMPACT-EMAP 2015, No. AS067-1, October 2015.
[2] K. Otsuka and Y. Sato, “Deep Learning Consideration with Novel Approach,” Proc. ICEP 2018, No. TD2-3, April 2018.
[3] D. Schneider, “Deeper and Cheaper Machine Learning,” IEEE Spectrum, pp. 38–39, Jan. 2017.
[4] K. Sato, C. Young, and D. Patterson, “An in-depth look at Google’s first Tensor Processing Unit (TPU),” Google Cloud Big Data and Machine Learning Blog, Friday, May 12, 2017.
[5] T. Imai, “Power Saving to 1/4 by 8bit,” Nikkei NE, pp. 14–15, June 2017.
[6] N. Hemsoth, “Inside Intel’s Strategy to Integrate Nervana Deep Learning Assets,” The Next Platform, November 22, 2016.
[7] T. Imai, “MCU strategy for AI,” Nikkei NE, pp. 16–17, July 2017.
Daisuke Ogawa received his B.Sc. degrees in Mathematics and Economics in 2013 from the University of Alabama at Birmingham in the USA, his M.Sc. degree in 2016 from Meisei University. Since 2017, he has been a doctoral student of the graduate school of computer information science at Meisei University. Currently, he is studying on simulation, analysis and design of high-speed logic/memory circuits using for FPGAs.

Yoichi Sato received his B.Sc. degree in 1975, his M.Sc. degree in 1977, all from Sophia University in Japan. From 1977–2010, he worked for Hitachi Ltd. He is currently a faculty member at the Collaborative Research Center of Meisei University.

Yasutaka Wada is an associate professor of the Department of Information Science at Meisei University, Tokyo, Japan. Before joining Meisei University in 2015, he was an assistant professor at Waseda University, Tokyo, Japan, an assistant professor at the University of Electro-Communications, Tokyo, Japan, a junior researcher at Waseda University, and a research associate at the same university. He was also an associate professor at Egypt-Japan University of Science and Technology (E-JUST), Alexandria, Egypt from 2010 to 2012. He received his B.Sc. and M.Sc. degrees in electrical engineering and Ph.D. degree in computer science and engineering from Waseda University.

He has served as a program committee vice chair of the IEEE COOL Chips conference series since 2015. His research interests include parallel processing and applications, green computing, heterogeneous computing, automatic parallelizing compilers, and multicore/manycore processor architecture. His research and development activities cover various systems from embedded systems to high-performance computing environments to realize high-performance, energy efficient, and easy-to-use computer systems. He is a member of ACM, IEEE Computer Society, IEICE, and IPSJ.

Kanji Otsuka received his B.Sc. degree in 1958 from Kyoto Institute of Technology, and his Ph.D. degree in 1987 from Tokyo Institute of Technology. From 1959 to 1993, he worked for Hitachi Ltd., where he had been engaged in the research and development of the semiconductor and general purpose/super computer technologies. From 1993–2008, he was a full professor of department of Electronic Information Science, faculty of Information Science, Meisei University. He is currently an executive researcher and emeritus professor of the Collaborative Research Center of Meisei University. He is a Fellow of IEEE.