Architecture of an FPGA-Based Brain Neural Network Simulator Using Direct Mapping

Hasitha Muthumala Waidyasooriya, Mizuki Harasawa and Masanori Hariyama
Graduate School of Information Sciences, Tohoku University
6-3-09, Aramaki-Aza-Aoba, Aoba, Sendai, Miyagi 980-8579, Japan
{hasitha@, harasawa.mizuki.q4@dc., hariyama@}tohoku.ac.jp

Abstract— Simulating brain neural networks is crucial for gaining insights into the functioning of the brain and for advancing the development of artificial intelligence. However, simulating large neural networks is very time consuming process. We propose an FPGA architecture to accelerate the simulation using parallel processing. Our proposed architecture employs a unified scheduling and allocation scheme to effectively increase the number of neurons while maintaining a high degree of parallelism. Our results demonstrate an impressive over 80% reduction in area without compromising on processing speed.

I. Introduction

Brain neural networks refer to a complex network of neurons in the human or animal brain. These biological neural networks are responsible for processing and transmitting information within the nervous system to achieve various body functions. Simulating neural networks is very important to understand how the brain works, and to develop artificial intelligence. Neuronal network simulation has gain much attention recently due to the advances in high performance computing (HPC) [1].

The Leaky Integrate-and-Fire (LIF) model [2] is a simplified mathematical model used to describe the behavior of individual neurons in the brain. It is a basic and widely used model in neuroscience for understanding the dynamics of neural membrane potentials and the generation of spikes. In this model, information transfer among neurons is represented by an electrical circuit with a parallel combination of a resistor and a capacitor. A current source is used as synaptic current input to charge up the capacitor to produce a potential. When neuron voltage exceeds a threshold, a spike is issued. To achieve a large scale neural network simulation, a power and area efficient implementation on computers is essential. FPGAs have significant potential for power-efficient neural network simulations.

The basic idea of direct mapping of neurons to FPGA hardware has been intorduced in previois studies [3]. While this method enables parallel execution and autonomous neuron control, it significantly restricts the FPGA's resource utilization, limiting its capacity to smaller-scale simulations.. This paper proposes a novel scheduling and allocation method to implementing large neural networks on FPGAs.

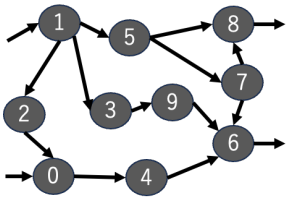


Fig. 1. Example of a brain neural network.

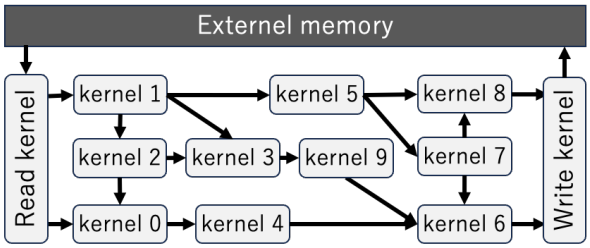


Fig. 2. FPGA architecture using direct mapping.

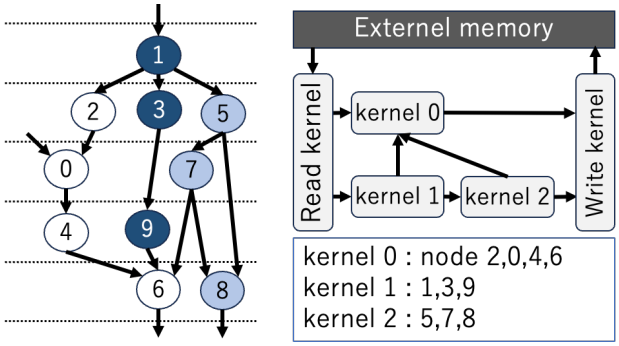
II. FPGA architecture using direct mapping

In Fig.2, we illustrate the direct mapping of neurons (nodes) to FPGA hardware. The initial potential data is stored in the external memory of the FPGA and is subsequently transferred to the respective kernels. Each individual node is represented by an OpenCL kernel, and Inter-kernel data transfer is done using OpenCL channels. A channel is a data transfer mechanism between two kernels based on handshake method. The read and write operations are efficiently managed by two dedicated kernels, while the remaining kernels are designed as "autorun" kernels, enabling automatic execution and control in response to data availability. This approach maximizes parallelism while minimizing control overhead. Despite allowing the maximum degree of parallelism, the utilization of dedicated kernels for each node significantly increases the resource utilization.

Table I shows the resource utilization of the direct mapping of neurons. The logic utilization is 73% to 100 neurons. Therefore, it is impossible to implement a large number of neurons. To address this challenge, we propose a novel scheduling and allocation scheme

TABLE I
COMPARISON OF RESOURCE USAGE.

Resource	Number of neurons	
	50	100
Registers	964,065	1,341,644
Logic	494,888 (53%)	679,384 (73%)
DSP	600 (10%)	679 (12%)
BRAM	632 (5%)	785 (7%)



(a) Scheduled DFG. Nodes with the same color are assigned to the same kernel.

(b) Proposed FPGA architecture.

Fig. 3. Scheduling of nodes and allocating nodes to kernels.

to implement a large number of neurons, without compromising on the processing speed.

We represent a brain neural network as a data-flow graph (DFG), where neurons are assigned to nodes, and information transfers are represented by edges connecting these nodes. We make an assumption that the processing time of every neuron is equal. Using this assumption, we apply ASAP (As Soon As Possible) and ALAP (As Late As Possible) scheduling on the DFG to determine the mobility of the nodes. Subsequently, we perform list-based scheduling and kernel allocation simultaneously. Our priority list is constructed based on the following criteria:

- We set a maximum limit on the number of kernels, which in turn restricts the level of parallel operations at each step.
- Nodes with smaller mobility are given higher priority.
- Nodes that are connected to each other are more likely to be assigned to the same kernel to optimize data transfers.

Fig.3a shows an example of the proposed method performed on the neural network in Fig.1. Nodes sharing the same color indicate that they can be assigned to the same kernels. The nodes scheduled in the same control step are assigned to different kernels to maximize parallel processing. On the other hand, nodes that are connected and scheduled to different control steps, are assigned to the same kernel to reduce resource utilization and to simplify the interconnection network. Fig.3b shows the accelerator architecture for this neural network. Excluding the read and write kernels, the number of kernels has been reduced from 10 to 3, and data transfer between kernels has been reduced from 13 to just 3 when compared to the previous architecture shown in Fig.2. This substantial reduction of resource utilization has been achieved without any compromise on processing time.

III. Evaluation

We have successfully implemented the proposed scheduling and allocation method using Python, enabling us to automatically generate FPGA architectures for handling more than 1000 neurons in less than a minute. The resource utilization can be reduced by over 80% compared to the previous studies [3] without a significant processing time penalty. The proposed accelerator is implemented on a BittWare 520N FPGA board, featuring a Stratix 10 GX FPGA. The FPGA kernel codes were compiled using Intel FPGA SDK for OpenCL version 19.4, while host codes were compiled using gcc version 10.2.0. Through our evaluation, we've demonstrated the feasibility of accommodating over 1000 neurons on a single FPGA.

IV. Conclusion

We introduce a novel scheduling and allocation scheme that substantially expands the scale of neural network simulations while incurring minimal processing time overhead. Our method enables the simulation of over 1000 neurons on an FPGA, with a high degree of parallel processing. This scheduling and allocation approach allows for the automatic generation of FPGA architectures in under a minute. Furthermore, for even larger neural network simulations, scalability can be achieved by employing multiple FPGAs or by adjusting the processing time constraint to strike a balance between scale and speed.

Acknowledgment

This research is partly supported by MEXT KAKENHI, grant number 20H04197.

References

[1] J. Jordan, T. Ippen, M. Helias, I. Kitayama, M. Sato, J. Igarashi, M. Diesmann, and S. Kunkel. Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Frontiers in neuroinformatics*, page 2, 2018.

[2] S. Dutta, V. Kumar, A. Shukla, N. R. Mohapatra, and U. Ganguly. Leaky integrate and fire neuron by charge-discharge dynamics in floating-body mosfet. *Scientific reports*, 7(1):1–7, 2017.

[3] Mizuki Harasawa, Hasitha Muthumala Waidyasooriya, and Masanori Hariyama. Direct Mapping of Neural Circuits on FPGA. In *23rd Int. Conf. on Parallel and Distributed Comp., Applications and Technologies (PDCAT'22)*, 2022.