# CELLULAR NEURAL NETWORK (CENN) FPGA IMPLEMENTATION USING MULTI-LEVEL OPTIMIZATION

*Zhongyang Liu[1], Shaoheng Luo[1], Xiaowei Xu[2] and Cheng Zhuo[1*]*
[1] Zhejiang University, Hangzhou 310058, China
[2] University of Notre Dame, Notre Dame, Indiana, 46655, USA
*Corresponding Author's Email: czhuo@zju.edu.cn

## ABSTRACT

Cellular Neural Network (CeNN) has been widely adopted in image processing tasks, which is considered as a powerful paradigm for embedded devices. Recently, digital implementations of CeNNs on FPGA have attracted researchers from both academia and industry due to its high flexibility and short time-to-market. However, most existing implementations are not well optimized to fully utilize the advantages of FPGA platform with unnecessary design and computational redundancy that prevents speedup. We propose a multi-level optimization framework for energy efficient CeNN implementations on FPGAs. In particular, the optimization framework is featured with three level optimizations: system-, module-, and design-space-level, with focus on computational redundancy and attainable performance, respectively. Experimental results show that our framework can achieve an energy efficiency improvement of 3.54× and up to 3.88× speedup compared with existing implementations.

***Keywords—FPGA; Cellular neural network; acceleration***

## INTRODUCTION

Cellular Neural Network (CeNN) is a nonlinear cellular processor array inspired by the functionality of neurons through modeling the working principles of human brain sensing. It has widely applied to various image processing areas such as noise cancellation, edge detection, path planning and segmentation [3]. Recently, CeNNs receive increasing interests from both academia and industry [5] for efficient hardware implementations.

CeNN accelerations on digital platforms such as ASICs [4], GPUs [6] and FPGAs [1][7] have been explored. Among them, FPGA has its unique potential and appears to be the most popular option due to its high flexibility and low time-to-market. With all these works sharing the same architecture and various acceleration techniques, there still lack a comprehensive study and design exploration for efficient CeNN implementation, which covers the following remaining issues:

- First, many existing works only employ a single processing element to compute one iteration, and hence do not exploit the full potential of parallelism.
- Second, many works are unaware of the repeated parameters in the template and incur unnecessary I/O overheads. Since I/O access is much slower than computation, such unawareness effectively leads to computational redundancy.
- The last but not the least, many design space exploration do not well study the utilization of available resources and bandwidth of FPGAs to achieve the best performance.

Thus, in this paper, in order to address those remaining concerns, we propose a multi-level optimization framework for CeNN computation for scalability and efficiency. The framework is featured with three-level optimizations:

- System level optimization (SLO): A parallel and tilling processing (PTP) and a data-reuse optimization are proposed to enable the parallelism of CeNNs.
- Module level optimization (MLO): For processing elements, a parameter quantization scheme is adopted. By eliminating the processing of repeated parameters, we can further simplify the computation process.
- Design space level optimization (DSLO): Based on available resources and I/O bandwidth, an optimal model about attainable performance estimations to explore the design space is proposed.

## FRAMEWORK OVERVIEW

As shown in Figure 1(a), the optimization framework is composed of external memory, memory interface controller, on-chip input and output buffers, a computation acceleration unit (CAU) and AXI4 bus. Due to on-chip resource limitation, data are stored in external memory and cached in on-chip buffers before processed in CAU. The workload is divided in temporal domain with a fully pipelined structure, where different iterations are processed simultaneously in different iteration units (IUs). The architecture is designed to load image stream from external memory, process it with CeNN and convert the result back. CAU constitutes the most important component, which is described in detail as follows:

- Iteration Unit A (IUA): The processor chain in CAU begins with IUA, which calculates the sharing parameters for all Euler iterations [7] in a time-invariant model. Thus only one IUA is implemented.
- Iteration Unit B (IUB): IUB indicates a number of identical modules in a fully-pipelined architecture. Pipeline depth is equal to the total number of Euler iterations desired.
- Data Transfer Controller: Data transfer controller manages the data transmission among the input buffer

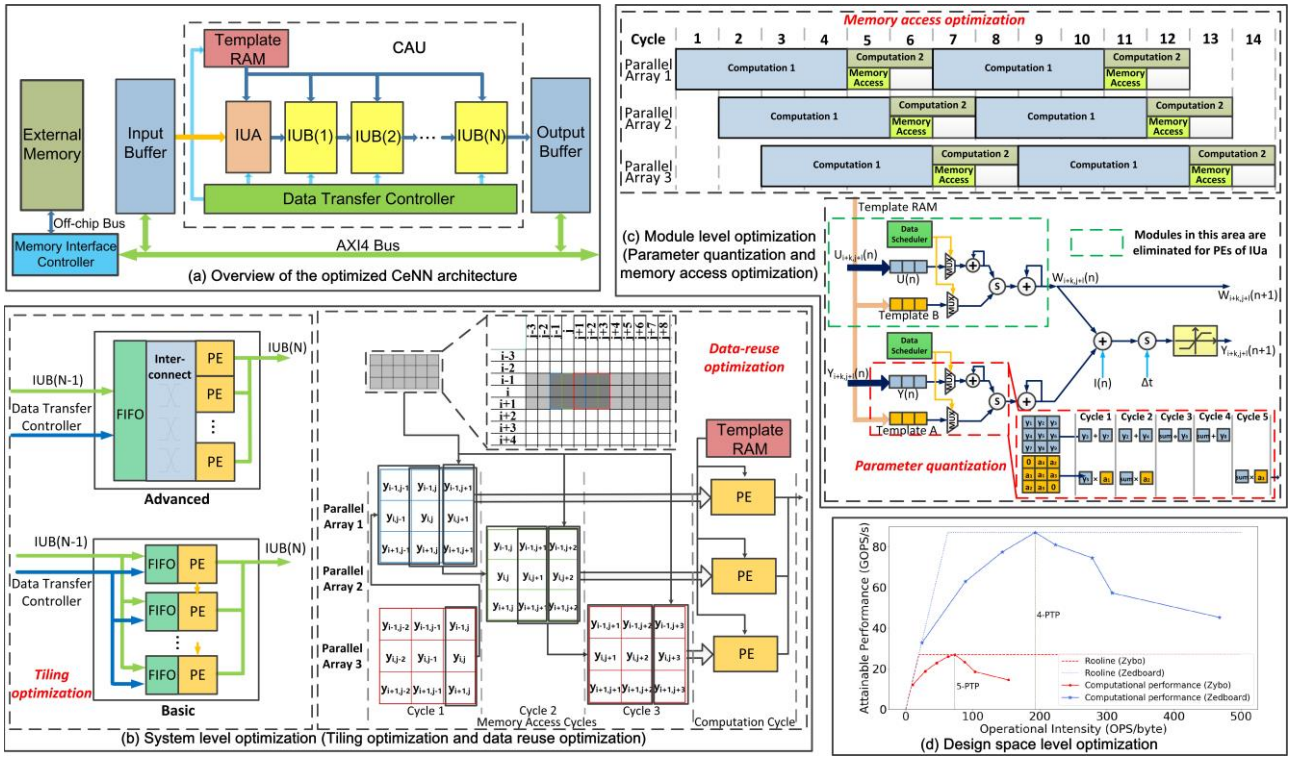


*Figure 1: (a) The optimized architecture and associated multiple-level optimization framework: (b) system level optimization, (c) module level optimization, and (d) design space level optimization.*

and the output buffer, template RAM and FIFOs in pipelined IUs. It also guarantees the synchronization of data transmission and computation in case of variant bandwidth features of the off-chip bus.

- Template RAM: The weights in CeNNs are space-invariant. Therefore, Template RAM needs only a few hardware resources for storage and memory access.

## MULTI-LEVEL OPTIMIZATION

In system level optimization (SLO), tiling optimization and data-reuse optimization are proposed in Figure 1(b). N-parallel and tiling processing (N-PTP) divides the workload into N parallel arrays for parallel operations and improves the throughput. PTP can be implemented in two strategies and allocate IUs in different structures resulting in different internal memory access operations. Input images are split into separate data chunks for advanced strategy. Data in each data chunk are arranged spatially with continuity and operated in one cycle, which reduces DRAM ports and hardware resources consumption compared with basic strategy. Data-reuse optimization utilizes repeated values in adjacent pixels, which eliminates excessive read and write operations of IUs and the input buffer. When calculations are executed in IUs, data-reuse based on data sharing relationships results in 2/3 memory access reduction.

In module level optimization (MLO), a larger number of multiplications are required using the limited embedded multipliers on FPGAs. Parameter quantization is proposed to reduce multiplier consumption by transforming multiplications into logic shifts, which enables repetition-induced and sparsity-induced optimization. It is found to be efficient for reducing multiplier consumption in most CeNN applications. Moreover, it is worthy to reorganize and manipulate the on-chip memory access and computation scheme to reduce processing latency. By investigating the memory access optimization in advanced PTP strategy as depicted in Figure 1(c), we overlap parallel phases, and arrange memory access cycles in a pipelined strategy. When no offloading operation is required in *Computation 2*, registers are prepared to load data for the next phase in memory access cycle.

Design space level optimization (DSLO) adopts the roofline model to recognize the peak performance for specific FPGA devices [2]. On one hand, a range of computational performance can be obtained by adopting optimization techniques. On the other hand, available resources and I/O bandwidth are two main limiting factors to CeNN. As shown in Figure 1(d), we explore the design space for the target platform and enumerate a set of implementations with the highest attainable performance. In most cases, we could find more than one design within the set, and the optimal solution can be selected for the given platform with desired performance and operational intensity.

TABLE I. RESOURCE UTILIZATION AND PERFORMANCE COMPARISON

| | Basic | SLO | MLO | SLO+ MLO |
|---|---|---|---|---|
| LUTs | 15336 | 10232 | 13632 | 7792 |
| FFs | 10824 | 6944 | 10656 | 6432 |
| Computational performance (GOPS/s) | 5.75 | 5.34 | 9.06 | 10.35 |
| Computational density (GOPS/LUT) | 3.75× E-4 | 5.22× E-4 | 6.65× E-4 | 13.28× E-4 |
| Improvement | 1× | 1.39× | 1.78× | 3.54× |

## RESULTS AND CONTRIBUTIONS

Table 1 summarizes the performance comparison of the basic CeNN approach (Basic) [7] and the implementation of the proposed framework using different optimization schemes. Giga operations per second (GOPS/s) is used as the metric of computational performance. For a fair comparison, the concept of computational density is introduced as a measure of energy efficiency, which is defined as the average giga operations per area unit (GOPS/LUT). The combined optimization technique SLO+MLO achieves a speedup of 1.8× and an energy efficiency improvement of 3.54× with its resource utilization decreasing to 50.8% compared with the basic approach, which is much larger than utilizing MLO or SLO alone.

DSLO is also discussed in Figure 2, where resource utilization is constraint and pipeline depth is variable in the same FPGA platform. The optimal model based on SLO+MLO+DSLO has the highest computational performance and the least pipeline depth. The highest computational performance achieves a 3.88x speedup.

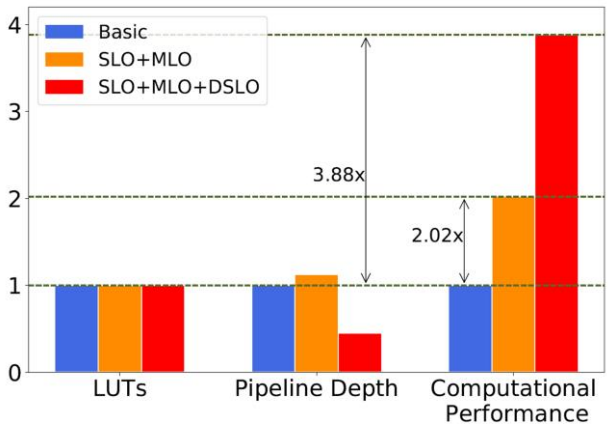


*Figure 2: Pipeline depth and performance comparisons among Basic, SLO+MLO and SLO+MLO+DSLO.*

## CONCLUSIONS

We propose an efficient multi-level optimization framework for the FPGA CeNN implementations. In system level, parallel and tilling processing (PTP) and data-reuse optimization are applied. The computation flow is improved by separating the workload in spatial domain. In module level optimization, we propose parameter quantization to reduce multiplier overheads and memory access optimization for the data transfer mechanism. In design space level, we apply different aforementioned optimized designs and obtain roofline models accordingly of attainable performance and operational intensity. Then the optimal CeNN solution can be selected with desired performance and operational intensity. Finally, evaluations of the proposed framework are presented on different FPGA platforms, and a speed up of 3.88× is achieved compared with existing implementations.

## ACKNOWLEDGEMENTS

The authors would like to thank the support from NSFC Fund (Grant No. 61601406).

## REFERENCES

[1] H. Chen, *et. al.* "Image-processing algorithms realized by discrete-time cellular neural networks and their circuit implementations." *Chaos, Solitons & Fractals*, 29(5), pp. 1100-1108, 2006.

[2] C. Zhang, *et. al.* "Optimizing fpga-based accelerator design for deep convolutional neural networks." *Proc. of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161-170, 2015.

[3] D. Manatunga, *et. al.* "Cellular neural network based medical image segmentation using artificial bee colony algorithm." *Proc. of International Conference on Green Computing Communication and Electrical Engineering*, pp. 1-6, 2014.

[4] S. Lee, *et. al.* "24-GOPS 4.5-mm$^2$ Digital Cellular Neural Network for Rapid Visual Attention in an Object-Recognition SoC." *IEEE Transactions on Neural Networks*, 22(1), pp. 64-73, 2011.

[5] D. Manatunga, H. Kim, and S. Mukhopadhyay. "SP-CNN: A Scalable and Programmable CNN-Based Accelerator." *IEEE Micro*, 35(5), pp. 42–50, 2015.

[6] S. Potluri, *et. al.* "CNN based high performance computing for real time image processing on GPU." *Proc. of International Symposium on Theoretical Electrical Engineering*, pp. 1-7, 2011.

[7] N. Yildiz, *et. al.* "Architecture of a fully pipelined real-time cellular neural network emulator." *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(1), pp. 130-138, 2015.