Application of Crop-Sum Algorithm to Character Recognition and Pedestrian Detection by Memory-Centric Computing

Ke Yu
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, South Korea
yuke12345@naver.com

Bobokhon Yusupbaev
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, South Korea
bobokhon9819@gmail.com

Minguk Kim
School of Electronics Engineering
Kyungpook National University
Daegu, South Korea
engh6542@naver.com

Jun Rim Choi*
School of Electronics Engineering
Kyungpook National University
Daegu, South Korea
jrchoi@ee.knu.ac.kr

***Abstract*—In the era of artificial intelligence, the popularity of portable devices and the development of Computer Vision (CV) have improved the convenience of human productive life. However, with the substantial increase of application data and the problems of traditional Von Neumann Computing architectures in terms of performance bottlenecks and power consumption becoming more and more prominent, there is an urgent need to propose new computing models and related optimization algorithms. Memory-Centric Computing (MCC) is considered as a good hardware option to solve this problem. This paper also presents a Crop-Sum algorithm that can be widely used for computer vision development and analyzes its feasibility for use in applications such as character recognition and pedestrian detection. The work results show that the text recognition application optimized using this algorithm is significantly improved in terms of performance and power consumption, and that the Crop-Sum algorithm is feasible for implementing MCC and computational optimization.**

***Index Terms*—Crop-Sum, Memory-Centric Computing, Optical Character Recognition, Pedestrian Detection, Computer Vision, Xilinx**

I. Introduction

The emergence and development of Artificial Intelligence (AI) has greatly contributed to the improvement of human productivity levels [1]. As an important branch of artificial intelligence, Computer Vision (CV) technology has an extremely wide range of applications, enabling computers to understand, analyze and interpret visual data from images or videos [2]. Computer vision aims to replicate and enhance human visual abilities using algorithms and models. In the field of Artificial Intelligence, data, algorithms and computing power are the three major elements of AI, which depend on and promote each other and together drive the rapid development of Artificial Intelligence technologies and applications. Computer vision algorithms process and extract meaningful information from visual data, enabling machines to recognize objects, understand scenes, and make intelligent decisions based on visual input.

The computing power of Artificial Intelligence is provided by computer devices. Traditional computing devices mainly rely on the Von Neumann architecture [3], where the computational and storage modules are separated from each other as shown in Fig. 1 (a). The data is usually stored in the memory, and in the process of computation, the data in the memory needs to be read into the processor first, and the result data needs to be returned to the memory again after the computation is completed. The advantage of the Von Neumann architecture is that it allows the components inside the computer, mainly the processor and memory, to develop independently of each other, and it has contributed to the high speed of computer development during the past decades.

However, as we entered the 21st century, the amount of data in the age of Artificial Intelligence began to increase dramatically [4]. Especially because of the popularity of smartphones and portable devices, the number of digital images and digital videos has grown more significantly. The growth of data has put higher demands on the computing power of computing devices. However, unfortunately, under the traditional Von Neumann architecture, the computing power of computing devices is slow to increase and obviously cannot meet the increasing data requirements. One is that the development speed of memory lags far behind that of processors, causing computational bottlenecks. Second, due to the significant increase in data in the application, the processor in reading and writing data from the memory when the energy consumed, has been much greater than the energy consumed in the processor to perform the calculation itself, forming a "Memory wall" problem. As the data is predicted to continue to increase significantly in the future, so the "Memory wall" problem may become more serious. Compared with other applications, computer vision itself is a highly data-centric field, so how to efficiently process the

* Correspoonding Author: Jun Rim Choi(jrchoi@ee.knu.ac.kr)

data becomes a very difficult problem. We need to consider the algorithm and computing device level to optimize the existing computing model.

In this paper, we propose a "Crop-Sum" algorithm based on a "Memory-Centric Computing (MCC)" optimization method. This algorithm can be widely used in the field of recognition in computer vision, such as character recognition [5] and pedestrian detection [6]. In our work, we have used the "Crop-Sum" algorithm to develop text recognition applications and explored its feasibility in pedestrian detection systems. We also experimented with a hardware implementation based on MCC using High Level Synthesis (HLS) and Field Programmable Gate Arrays (FPGAs). The experimental results show that the optimized system using this algorithm has significant advantages over traditional algorithms..

The remaining sections of this paper are organized as follows: Section II presents the background. The Crop-Sum algorithm and applications are introduced in Section III. Section IV describes the software and hardware implementation of OCR system application and Pedestrian Detection. Section V discusses the results of implementation. Section VI presents the conclusions of this paper.

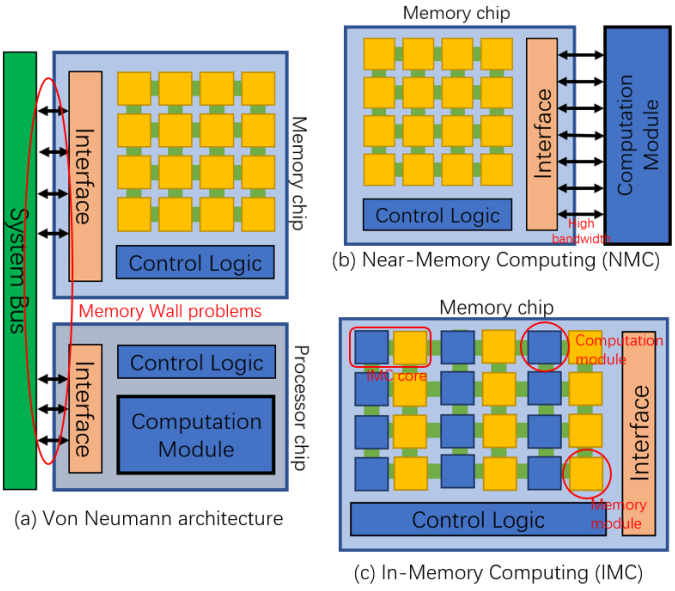

Fig. 1. Comparison of (a) Von Neumann architecture, (b) Near-memory Computing (NMC) and (c) In-memory Computing (IMC).

II. Background

Memory-Centric Computing (MCC) is considered an effective way to solve the performance bottleneck and "Memory wall" problem of computing under the Von Neumann architecture. Unlike the traditional Von Neumann architecture of Processor-Centric Computing, MCC focuses on shortening and reducing the data transfer between the processing and storage modules so that part of the computation can be performed without the need to go through the processor. By packaging the processing logic near the memory, the data movement distance can be shortened by performing Near-memory Computing (NMC) as shown in Fig. 1 (b). Alternatively, the architecture of conventional memory can be directly changed so that memory can read and write data while performing a portion of the computation previously required by the processor, called In-memory Computing (IMC) as shown in Fig. 1 (c). This reduces the burden on the processor and naturally reduces the energy waste caused by data movement.

Currently, MCC has been extensively studied in academia. Since MCC is a Near-data computing and the computational cores in memory cannot be designed as large as those in processors, MCC is more suitable for computations with high data volume but low complexity. As more than half of the current applications in AI rely on Multiplication and Accumulation (MAC) and Matrix-Vector Multiplication (MVM) computations [7], most researchers hope that by changing existing memory architectures and arrays, or investigating new types of memories such as MRAM (Magnetic Random-Access Memory), PCRAM (Phase Change Random-Access Memory), and RRAM (Resistive Random-Access Memory), so that memories can perform MAC and MVM computations while reading and writing data, this will be very effective in reducing the processor's burden in performing AI applications.

On the other hand, since MCC breaks the traditional way of computation, it is also important to study the dedicated algorithms for MCC according to the characteristics of various AI applications [9]. In particular, as memory is difficult to perform complex computations, the implementation of MCC will be facilitated by the development of specific algorithms that reduce the computational complexity of the application, as well as the amount of data to be processed.

III. Crop-Sum Algorithm and its Applications

In our previous work, we found that the Crop-Sum algorithm can often be used very effectively for computer vision system development in recognition applications. Especially, it can simplify the processing of data in images by means of pixel counting. This is in line with the MCC development expectations mentioned in Section II. In this section, we will specifically describe the principles of operation of the Crop-Sum algorithm and the exploration of the algorithm in terms of practical applications.

A. Description of Crop-Sum Algorithm

The Crop-Sum algorithm is used to calculate the sum of pixels for a specific rectangular area in the input image. Fig. 2 shows the operation principle of the Crop-Sum algorithm. In computer vision for binary images, black pixels have a value of 0 and white pixels have a value of 255. As shown in Fig. 2, with the Crop-Sum algorithm, we can sum the pixel values of a 3×3 ROI (Region of Interest) [10] area in the center of an input image of size 9×9, and the result of Crop-Sum is 1020.

By using the Crop-Sum algorithm to perform pixel calculation on some ROI regions in an image, we can get a lot of information related to the image. In our research, we found that many digital images and videos in daily life have special pixel regularity. For example, the strokes in characters only appear in specific positions, like the middle of "O" and "0", which have no strokes. Also, when the car is moving at a normal constant speed, the pixel change in

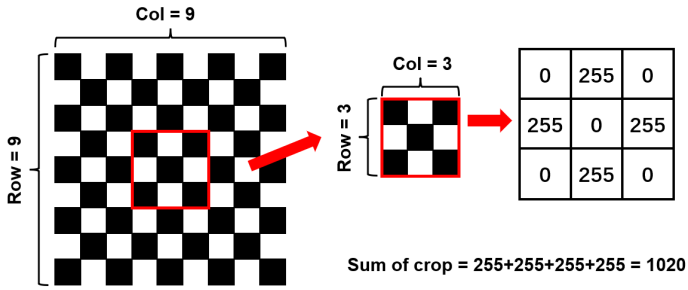

Fig. 2. The operation principle of the Crop-Sum algorithm.

the front view is not obvious. Therefore, we believe that we can optimize the computation process by implementing the Crop-Sum algorithm for specific ROI regions in the image based on this feature, which can be used in the development of character recognition and pedestrian detection systems.

B. Applications of Crop-Sum Algorithm

1) Optical Character Recognition:

Traditional optical character recognition (OCR) systems typically use a template matching method to discriminate characters by direct pixel similarity comparison between the input image and a standard template image. The similarity between them is calculated using a specific similarity calculation algorithm, and the template with the highest similarity will be recognized as the target. This algorithm is straightforward and requires no special design, but is computationally intensive and unstable in terms of data volume.

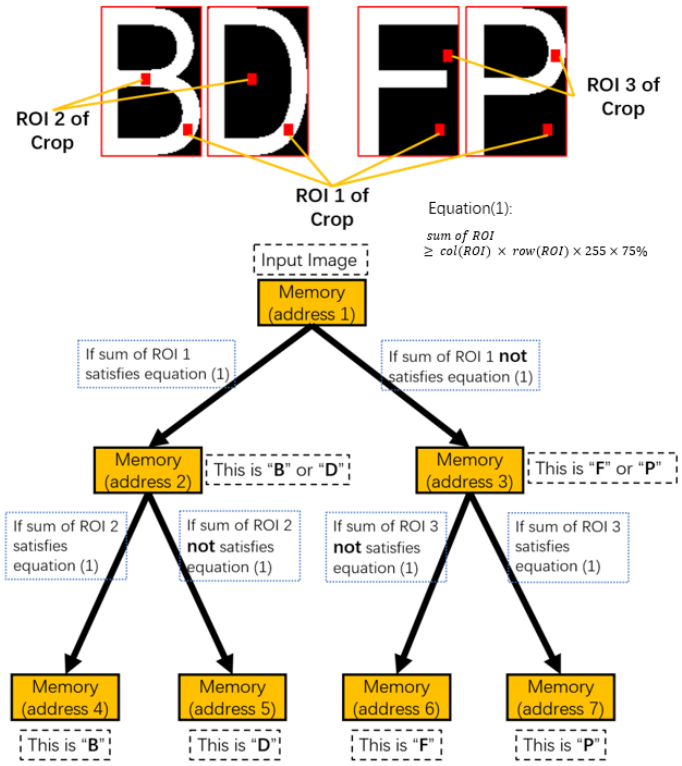

Fig. 3. Description of the core principle of the Character Recognition process based on Crop-Sum optimization with the example of distinguishing "B", "D", "F" and "P".

In this paper, we present a "Tree-Form" optimization design based on Crop-Sum algorithm to classify characters by the pixel similarity of their strokes, and we calculate the pixel values of specific ROI regions of characters several times to achieve character recognition. As shown in Fig. 3, characters "B" and "D" have the same pixel characteristics in ROI1, and characters "F" and "P" also have the same pixel characteristics in ROI1. When distinguishing "B","D","F" and "P", we can first calculate the pixel value of ROI1 to determine which group the characters belong to. For example, a character may belong to the "B and D" group, or a character may belong to the "F and P" group. Then we can calculate ROI2 or ROI3 to distinguish exactly which character.

The algorithm is characterised by the need to pre-calculate the character recognition process and design the hardware architecture based on the pre-calculated results. Due to the prior design of the data transmission process, the system avoids possible complex and cumbersome calculations during the actual operation. The use of the algorithm can be extended to perform recognition of various characters such as numbers and English letters.

2) Pedestrian Detection:

In addition to optical character recognition, we are also exploring the application of Crop-Sum algorithm in the field of autonomous driving and pedestrian detection. As shown in Fig. 4, vehicles need to keep distance from surrounding pedestrians during driving, i.e., they need to use pedestrian detection systems to keep driving safe. However, the safe environment occupies the vast majority of the driving process, and the front screen changes very little when the vehicle is moving at a constant speed. For example, black roads and distant skies with almost constant pixels. In the traditional computing mode, it is necessary to repeatedly calculate similar images, which will cause energy wastage, so it needs to be considered for optimization.

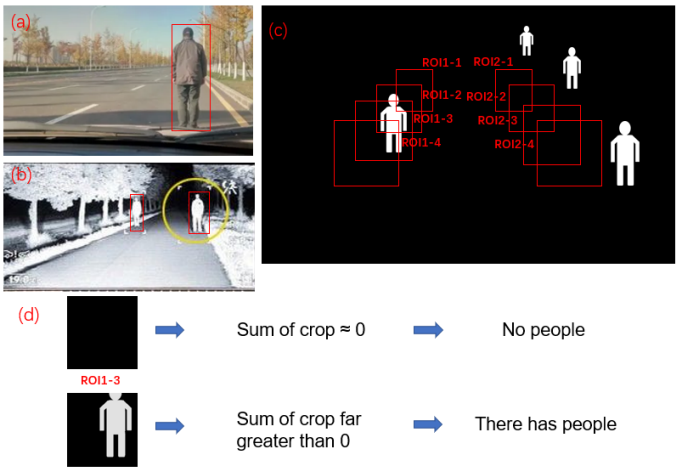

Fig. 4. (a) Pedestrian detection in RGB color image, (b) Pedestrian detection in hermal imaging image, (c) A model that mimics thermal imaging for pedestrian detection and (d) Crop-Sum algorithm for pedestrian detection.

We are trying to examine this using thermal imaging images taken after the infrared camera. The normal temperature of the human body is 35°C to 38°C. We can enhance this part of the display in the image and exclude other environmental

disturbances as much as possible. Analyze the area where pedestrians appear with high frequency and set that area as a vigilance area, such as a series of ROI areas shown in the Fig. 4 (c). The pixel sum of each vigilance region is obtained by executing multiple Crop-Sum algorithms on the vigilance regions. When the pixel sum changes significantly it is determined that pedestrians are present in the vigilance area and immediate stopping is required. On the contrary, there is no abnormality and the vehicle can continue to drive normally. The advantage of this design will be more obvious by performing parallel computation under hardware implementation conditions [11], which is expected to give good results in terms of execution time.

IV. Implementation and Results

A. Software Implementation of OCR

We first designed a character recognition system based on the Crop-Sum algorithm and the method shown in Fig. 3 that can recognize English uppercase letters and numbers. The architecture is shown in Fig. 5. Since C/C++ has a better correlation with hardware, we developed the design OCR system using C/C++ and computer vision functions from the OpenCV library for the software side of the implementation and tested the execution time. The platform parameters used in the OCR software implementation are shown in TABLE I. For comparison, we also developed the OCR system using the traditional template matching method mentioned in section III and tested it under the same conditions. The comparison results are shown in Fig. 6 and TABLE II.

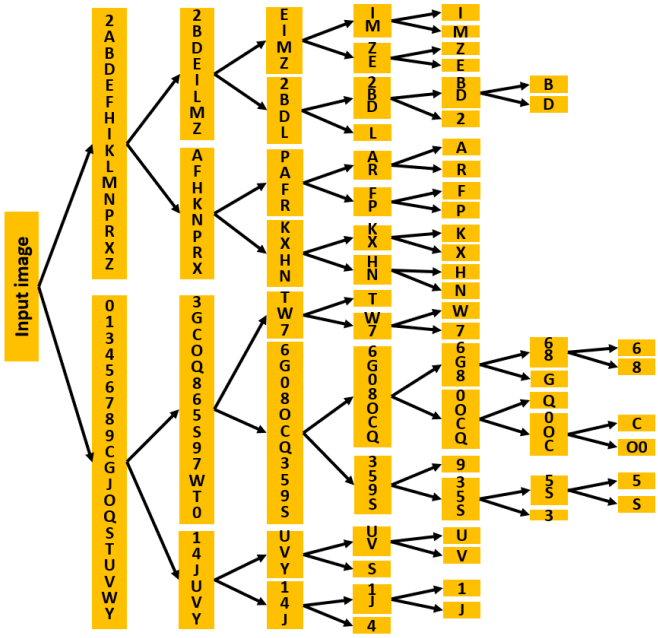

Fig. 5. Diagram of the "Tree-Form" structure of the Crop-Sum algorithm optimization for digits and English capital letters recognition.

In addition to the execution time, we also analyzed the memory accesses of both algorithms. The results were tested using the "Valgrind" tool under the TABLE I platform and are shown in TABLE III.

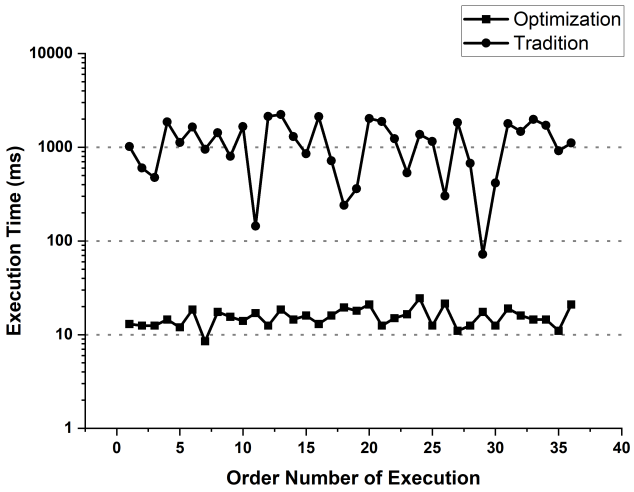

Fig. 6. Comparison with Conventional design and optimization design in software implementation.

TABLE I
Parameters of the computing system hardware and input image used for testing during software implementation.

	Processor	*Frequency*	*Power*	*RAM*
Parameter	Intel i7-4790	3.60GHz	84W	16GB
	Operating System	***Input image***	***ROI***	
Parameter	Ubuntu 18.04	80×120	6×6	

The results of the software implementation show that the OCR system design optimized using Crop-Sum has more significant advantages.

B. Hardware Implementation of OCR

In order to explore the hardware feasibility of Memory-Centric Computing, we have used the Xilinx Alveo U50 Accelerator and Vitis development Platform for the hardware implementation [12]. The Alveo U50 Accelerator is a high-performance FPGA that includes High Bandwidth Memory (HBM) and can be connected to an x86 CPU in a host PC via a PCIe interface to form a heterogeneous computing combination of processing system (PS) and programmable logic (PL).

We used the "Crop" and "Sum" functions from the "Vitis Accelerated Library" and designed the same algorithm as the

TABLE II
Average execution time of traditional design and optimization design.

	Traditional Design	*Optimization Design*
Average execution time	1171.97 ms	15.46 ms

TABLE III
Comparison with traditional design and optimization design in memory accesses during software implementation.

	Traditional Design	*Optimization Design*
Memory usage (bytes)	2,827,448	89,734

software implementation for the hardware implementation. The designed optimized character recognition system was transformed by Xilinx Vitis HLS to form the PL part [13]. And the PS part running on x86 CPU was compiled using C/C++ and OpenCL (Open Computing Language). The PL and PS parts communicate with each other based on the PCIe physical interface and the Xilinx Runtime (XRT) driver.

TABLE IV
PERFORMANCE AND POWER CONSUMPTION ESTIMATES FOR THE PL HARDWARE IMPLEMENTATION OF OCR OPTIMIZATION DESIGN.

FPGA Frequency	*Latency*	*Execution Time*	*FPGA Power*
300.300 MHz	10,273 Cycles	34.24 us	18.59W

TABLE V
UTILIZATION FOR THE PL HARDWARE IMPLEMENTATION OF OCR OPTIMIZATION DESIGN.

	BRAM_18K	*DSP*	*FF*	*LUT*
Total of use	234	0	36,663	72,813
Available	2688	5952	1,743,360	871,680
Utilization	8%	0%	2%	8%

Our design was successfully synthesized and implemented on this platform and completed running, verifying the feasibility of the design in terms of hardware. TABLE IV and V show the performance estimates and resource utilization of the PL part of the hardware implementation, respectively. Comparisons with other similar work are shown in TABLE VI.

TABLE VI
COMPARISON WITH RELATED WORK IN HARDWARE IMPLEMENTATION OF CHARACTER RECOGNITION.

	This work	**[5]**	**[14]**
Platform	Xilinx Alveo U50	Xilinx Zynq-7000	Xilinx Virtex IV FPGA
Type of characters	English capital letters and Arabic digits	Arabic digits	English capital letters and Arabic digits
Frequency	300.300 MHz	114.416 MHz	N/A
Pixel size	80 × 120	22 × 34	N/A
Execution time	34.24 us	0.63 ms	N/A
BRAM	234	10	N/A
DSP	0	20	N/A
FF	36,663	4,247	43,551
LUT	72,813	5616	50,310
URAM	0	N/A	N/A
Utilization	8%	6%	N/A

C. Hardware Implementation of Pedestrian Detection

For the pedestrian detection system, we have designed the hardware based on the principles in Fig. 4 (c) and (d), also using the HLS method. The designed hardware was successfully synthesized in C under Vitis HLS tool and passed the RTL simulation. The test procedure was performed using Fig. 4 (c) with input image size 1800 × 1100 (Width × Height). In the RTL simulation session, the pixel sums of the eight ROI regions in the image were calculated accurately.

Table VII shows the performance estimates of the developed pedestrian detection application of the hardware implementation under Vitis HLS, and Table VIII shows its hardware resource utilization.

TABLE VII
PERFORMANCE ESTIMATES FOR THE HARDWARE IMPLEMENTATION OF PEDESTRIAN DETECTION.

FPGA Frequency	*Latency*	*Execution Time*
300 MHz	2,186,753 Cycles	7.216ms

TABLE VIII
UTILIZATION FOR THE HARDWARE IMPLEMENTATION OF PEDESTRIAN DETECTION.

	BRAM_18K	*DSP*	*FF*	*LUT*
Total of use	50	0	19,466	29,531
Available	2,688	5,952	1,743,360	871,680
Utilization	1%	0%	1%	3%

V. DISCUSSION

In optical character recognition system, since the optimized design using the Crop-Sum algorithm can control the computation mainly on the summation of pixels in the crop range, thus avoiding multiple iterations of all pixels of the input character image, the design is also superior to the conventional algorithm purely in terms of software. The results in Fig. 6 and TABLE II show that the optimized design is about 75 times faster than the traditional design in terms of software execution time. At the same time, the test results of the "Valgrind" tool show that the efficiency of the processor in reading and writing data to the memory during the execution of the software is much better than that of traditional algorithms.

The results of the hardware implementation of OCR show that the design can recognize a single character in 34.24us and that the hardware executes with about 18.59W of power, which saves 77.87% of energy compared to the 84W of power required by the CPU. The optimized design based on the hardware implementation achieves faster computation with lower power consumption. The comparison in TABLE VI shows that despite the fact that we have increased the image size in order to maintain 100% recognition accuracy, resulting in a significant increase in the hardware resources used, the design still maintains a significant advantage in terms of performance.

For the development of pedestrian detection, although it passed the RTL simulation test of Vitis HLS, the effectiveness needs to be further verified. We have only used modelled binary images for testing, so the development results lack practical relevance yet. Because the real driving environment is much more complex than the model, we need to develop more correlations with the real environment. And the design has more overlapping parts in the selection of ROI regions and very larger image size, which will lead to computational inefficiency and unnecessary resource waste. The design still has high space for continuous optimization in the future.

Our work on OCR applications verifies the advantages of the Crop-Sum and MCC-based algorithm design in terms

of power consumption and computational performance, and demonstrates that the design is feasible in hardware. Also our work in pedestrian detection can strongly support the expansion of the application area of this algorithm. In the future, we will continue to verify the effectiveness of the Crop-Sum algorithm for pedestrian detection applications and explore more additional application scenarios of the algorithm in MCC-based applications. We will also try to use other implementations than HLS and FPGA to more fully verify its hardware reliability. At the same time, we also note that the processor still has an important role in the current computing model, and the traditional Von Neumann architecture is difficult to be completely replaced in a short time. In particular, MCC still lacks sufficient capability in performing complex computations and lacks sufficient verification of stability and reliability. Therefore, according to the characteristics of the application, it may be more ideal to design an architecture based on a combination of MCC and traditional computing models.

VI. Conclusions

In this paper, we discuss that a solution to the computational performance bottleneck and power consumption problems of traditional architectures is to adopt a new hardware architecture of Memory-Centric Computing (MCC) and to develop optimization algorithms corresponding to the characteristics of the application. We propose a Crop-Sum algorithm and analyze its feasibility for optimization of computer vision applications such as character recognition and pedestrian detection. The test results from the development of character recognition application show that the application optimized with the Crop-Sum algorithm has 75 times better performance in software than the traditional algorithm. After an MCC-based hardware implementation using an ALveo U50 FPGA, the system can recognize numbers or English uppercase letters in 34.24 us with FPGA power consumption of 18.59W, which is a significant performance and power advantage over computation performed by a CPU consuming 84W. The pedestrian detection hardware based on the Crop-Sum design was also successfully simulated. Our work shows the feasibility of the Crop-Sum algorithm for implementing MCC and computing optimization.

Acknowledgment

This work was supported by Samsung Electronics Co., Ltd. This study was also supported by the BK21 FOUR project funded by the Ministry of Education, Korea (4199990113966).

References

[1] A. Negi and K. Rajesh, "A Review of AI and ML Applications for Computing Systems," 2019 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19), Nagpur, India, 2019, pp. 1-6.

[2] Zehang Sun, G. Bebis and R. Miller, "On-road vehicle detection: a review," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pp. 694-711, May 2006.

[3] John Backus. 1978. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. Commun. ACM 21, 8 (Aug. 1978), 613–641.

[4] Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu. 2018. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18). Association for Computing Machinery, New York, NY, USA, 316–331.

[5] Farhat, A., Hommos, O., Al-Zawqari, A. et al. Optical character recognition on heterogeneous SoC for HD automatic number plate recognition system. J Image Video Proc. 2018, 58 (2018).

[6] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 4, pp. 743-761, April 2012

[7] A. Ankit, I. Chakraborty, A. Agrawal, M. Ali and K. Roy, "Circuits and Architectures for In-Memory Computing-Based Machine Learning Accelerators," in IEEE Micro, vol. 40, no. 6, pp. 8-22, 1 Nov.-Dec. 2020.

[8] Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. et al. Memory devices and applications for in-memory computing. Nat. Nanotechnol. 15, 529–544 (2020).

[9] Yu, K.; Kim, M.; Choi, J.R. Memory-Tree Based Design of Optical Character Recognition in FPGA. Electronics 2023, 12, 754.

[10] Iqbal, O.; Muro, V.I.T.; Katoch, S.; Spanias, A.; Jayasuriya, S. Adaptive Subsampling for ROI-Based Visual Tracking: Algorithms and FPGA Implementation. IEEE Access 2022, 10, 90507–90522.

[11] Stone, J.E.; Gohara, D.; Shi, G. OpenCL: A parallel programming standard for heterogeneous computing systems. Comput. Sci. Eng. 2010, 12, 66–72.

[12] Xilinx. Vitis Unified Software Platform Documentation: Application Acceleration Development. ug1393. 2022, 2.

[13] Xilinx. Vitis High-Level Synthesis User Guide. ug1399. 2022, 2.

[14] Caner, H.; Gecim, H.S.; Alkar, A.Z. Efficient embedded neural-network-based license plate recognition system. IEEE Trans. Veh. Technol. 2008, 57, 2675–2683.