HePiLUT: Resource Efficient Heterogeneous Pipelined CNN Accelerator for FPGAs

Rashed Al Amin
Institute for Embedded Systems
University of Siegen
Siegen, Germany
rashed.amin@uni-siegen.de

Md Shahi Amran Hossain
Institute for Embedded Systems
University of Siegen
Siegen, Germany
md.hossain@uni-siegen.de

Roman Obermaisser
Institute for Embedded Systems
University of Siegen
Siegen, Germany
roman.obermaisser@uni-siegen.de

***Abstract*—Field programmable gate arrays (FPGAs) have gained recognition as a suitable platform for implementing Convolutional Neural Network (CNN)-based algorithms due to their favorable attributes, such as energy efficiency and parallel computing capabilities. Nevertheless, deploying CNNs on resource-constrained devices poses challenges due to the substantial computational demands of CNNs and limited on-chip hardware resources. A majority of prior research predominantly relies on the limited block RAM (BRAM) for storing CNN parameters, and costly digital signal processing (DSP) blocks for executing multiply-accumulate (MAC) operations. To address these constraints, this paper introduces HePiLUT, a novel compiler designed to analyze the CNN structure and parameters, and automatically generates a heterogeneous pipelined CNN accelerator architecture by utilizing Look-Up Table (LUT). In addition, a dataflow controller in the pipelining architecture has been developed to foster a resource-efficient and low-latency CNN accelerator for FPGAs. HePiLUT leverages LUTs to store input image data and execute MAC operations instead of DSP blocks. The efficacy of the proposed CNN accelerator compiler has been evaluated on the Xilinx ZCU102 FPGA platform utilizing the Visual Geometry Group (VGG) architecture and CIFAR-10 dataset. The findings underscore HePiLUT's capability, showcasing a substantial reduction compared to state-of-the-art CNN accelerator design. However, these substantial savings come at a trade-off, as there is a 3% decrease in throughput performance. The proposed HePiLUT significantly reduces the FPGA resource utilization and automatic compiler solution in CNN implementation, providing considerable support for sustainable and energy-efficient computing systems.**

Keywords—CNN accelerator, heterogenous pipelining, FPGAs

I. INTRODUCTION

Deep Neural Networks (DNN), particularly CNN offer a robust method for extracting abstract features from image data, showcasing significant success in various computer vision applications [1] due to their achieved accuracy like humans [2]. CNNs typically comprise multiple layers involving computationally intensive convolution operations followed by classification layers, posing challenges for achieving real-time performance even on the most advanced computing platforms. However, this method demands substantial network bandwidth and prolonged processing periods, making it impractical for scenarios with stringent real-time requirements. Consequently, contemporary edge AI solutions frequently leverage FPGAs to implement and execute machine learning algorithms. FPGAs exhibits a LUT-based product term structure and superior parallel computing capabilities, contributing to enhanced energy efficiency and lower power consumption than alternatives like GPUs, CPUs, or VPU systems. Despite these advantages, CNN accelerators encounter limitations in memory bandwidth and capacity, a direct result of the CNN algorithms' computation and the memory-intensive nature, especially in resource-constrained edge devices like FPGAs.

Two common challenges encountered during the deployment of CNNs on FPGAs are related to computational intensity and memory intensity [3]. Researchers have proposed diverse CNN models and methodologies targeting these issues in response. Techniques aiming to alleviate the computational complexity of CNNs include sparse neural networks [4], low-precision fixed-point quantization [5], tensor decomposition [6], zero-transfer skipping methods [7], weight pruning [8], binarized neural networks [9], depthwise separable convolution [10], and the utilization of block convolutions to reduce computational load [11].

In tandem with these innovations, architectural innovations in memory management have been pursued, as contemporary CNNs tend to be larger in scale and cannot solely rely on on-chip memory for storage. Furthermore, novel computing architectures, such as 'non-von Neumann' paradigms [12], have been introduced, encompassing processing-in-memory (PIM) [13], in-memory computing (IMC) [14], near-data processing (NDP) [15], and compute-in-BRAM [16] all aimed at mitigating data transfer bottlenecks. Notably, LUT-based PIMs have arisen as a promising solution to enhance processing speed [17]. Besides, recent reports have also highlighted the effectiveness of LUT-based neural network architectures [18, 19] and LUT-based multiplier [20] for CNN applications.

Numerous methodologies have been explored to create resource-efficient CNN accelerators. Regrettably, the FPGA research community currently lacks a dedicated CNN accelerator compiler for hardware implementation of CNNs, which automatically generates a resource-efficient CNN accelerator based on input neural network description files. This paper introduces HePiLUT, a novel heterogeneous pipelining technique based compiler aimed at mitigating computational latency and resource consumption omitting the need for data storage within memories and foregoing the loading of complete image data. Evaluation of HePiLUT has been conducted using the Xilinx ZCU102 FPGA board, resulting in a performance of 129600 FPS for 32×32 pixel RGB images while consuming significantly fewer resources compared to the state-of-the-art CNN accelerator. The novel contributions of this paper are outlined as follows:

1. HePiLUT introduces a novel CNN accelerator compiler that generates a heterogeneous pipelined CNN module based on the input CNN architecture. This enhancement facilitates the orderly execution of parallel instructions across all CNN layers and RGB channels.

2. HePiLUT optimizes CNN accelerator design layer by layer, prioritizing resource efficiency and flexibility. By using LUT-based MAC operations, it eliminates the need for

979-8-3503-5469-0/24/$31.00 ©2024 IEEE

external memory access, BRAM, and DSP, streamlining the architecture for improved performance.

3. The performance evaluation of the acceleration system generated by the proposed HePiLUT CNN accelerator compiler has been conducted by comparing it with that of state-of-the-art CNN accelerators.

II. Related Work

Numerous techniques has been proposed to developed efficient CNN accelerators, aiming to improve computational power, reduce energy consumption, and optimize resource utilization. This section gives an overview of prominent architectures and methodologies reported in contemporary literature to identify state-of-the-art CNN accelerators.

Ma et al. [21] reported on a scalable and modularized RTL compilation of CNN for FPGA, which automatically generates a set of modular and scalable computing primitives capable of accelerating various deep learning algorithms. However, this method does not consider any pipelining architecture, resulting in comparatively lower throughput compared to recent CNN accelerators such as FINN [9], FixyFPGA [22], and HPIPE [23]. Several other CNN accelerator designs, such as HPIPE [23] and Tomatto [24], have been demonstrated and evaluated with 224×224 pixel images, exhibiting very high throughput. However, these designs also necessitated very high FPGA resources and did not consider accuracy.

Umuroglu et al. [9] introduced FINN, a scalable and fast CNN accelerator rooted in a binarized neural network framework.. FINN exhibits impressive performance by demonstrating a maximum classification speed of 21,906 images per second for an image size of 32×32 pixel images. However, despite its strengths, BNNs encounter limitations in leveraging the inherent FPGA architecture with high precision.

Meng et al. [22] introduced FixyFPGA, a CNN accelerator harnessing the element-wise sparsity technique for both VGG-7 and MobileNet. This method encodes the trained CNN network's weights directly into the hardware, utilizing them as fixed operands for multiplication. This approach achieves notable speeds by employing a fully parallel and pipelined computation engine, reaching 134,000 FPS for 32×32 pixel images.

The predominant focus of prior research has centered on DSP-based MAC operations for computation and BRAM to store the CNN weights, often resulting in high power consumption and resource demands. Despite the high performance offered by DSP blocks, their utilization may not always be optimal considering overall performance and area requirements, especially for large CNN applications due to limitations such as quantity, flexibility, clock frequency constraints, and restricted precision operations are associated with DSP blocks. In contrast, HePiLUT introduces a LUT-based heterogeneous pipeline architecture that leverages FPGA's inherent features of low latency and minimal resource utilization. Moreover, efficient control of the dataflow in the proposed architecture helps in reducing the number of LUT usage in FPGA fabric.

III. Implementation

The HePiLUT CNN accelerator has been implemented using a universal CNN framework based on the Very High-Speed Integrated Circuit Hardware Description Language (VHDL) for image classification tasks. Notably, HePiLUT is designed to cater to various FPGAs, with a particular focus on low-density FPGAs characterized by constrained memory and computational resources. The reconfigurable nature of LUTs enables support for different precision data, necessitating fewer LUTs for operations with reduced precision. Fig. 1 illustrates the block diagram of the proposed HePiLUT architecture. HePiLUT consists of three clusters: first, data preprocessing through software tools such as TensorFlow or PyTorch; second, the HePiLUT compiler, responsible for generating the CNN accelerator hardware design from the input neural network model; and third, synthesis and verification of the generated accelerator using Vivado.

A. Data Pre-Processing

The data pre-processing module plays a pivotal role in facilitating the interface between the CNN model and the HePiLUT CNN accelerator compiler. In this integral step, the input trained and quantized CNN model is required to conform to the .tflite format, which serves as the standardized medium for conveying crucial information pertaining to the neural network description and the underlying feature maps of the input image. Such meticulous inclusion ensures the seamless integration of essential model parameters and input data representations, facilitating the subsequent compilation and optimization processes within the HePiLUT architecture.

B. HePiLUT CNN Accelerator Compiler

The network analyzer module within the HePiLUT CNN accelerator compiler undertakes an initial investigation into the input CNN layers and their associated parameters. This analytical process encompasses the examination of critical attributes such as kernel size, stride size, and filter number across all convolutional and max-pooling layers. Subsequently, the generated accelerator design and its interconnections in the hardware are adjusted accordingly to align with these parameters. Additionally, HePiLUT's network analyzer evaluates user-defined computing pipeline

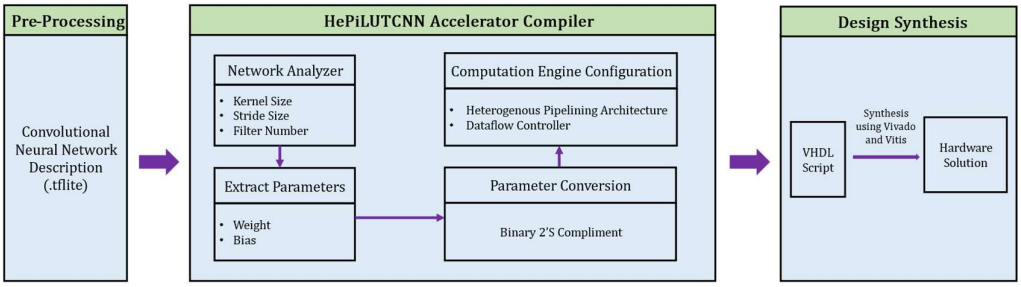

Fig. 1. Block diagram of the HePiLUT architecture.

architectures, thereby generating and integrating parameterized CNN modules. Moreover, the network analyzer orchestrates the synchronization of the clock to harmonize with the specifications of the FPGA device. In a bid to conserve power during calculations and curtail resource consumption, all parameters, including filters and weights for each respective layer, have been extracted from the trained model and converted into 8-bit fixed-point data. Notably, the optimization achieved through the use of 8-bit precision is deemed sufficient to maintain an acceptable level of inference accuracy. Following this synchronization, the parameters of the input CNN model, including weight and bias, are extracted and stored in a VHDL array format subsequent to conversion into binary 2's complement representation. This meticulous process ensures the computational engine configuration of the HePiLUT compiler.

The core of the proposed HePiLUT CNN accelerator compiler is its computation engine configuration, comprising interconnected blocks organized in a cascade configuration. These blocks demonstrate a heterogeneous pipeline structure, serving all layers of the CNN and handling the three RGB layers of the input image. The HePiLUT pipeline structure involves three key procedures—data load, data calculation, and data transfer to optimize the accelerator's efficiency.

1. Data Load

To mitigate the reliance on the slower Off-chip memory, a strategy has been employed wherein the input feature maps undergo processing without being stored in memory. The input image data is accessed from DDR memory by HePiLUT computational engine through Advanced eXtensible Interface (AXI) Stream. Loading of the input image data occurs in a line-by-line fixed row and column manner, populating the 3D array of input features and kernels stored in the LUT memory. However, the loading process for each CNN layer follows a specific arrangement. For the initial convolution layer of the CNN, HePiLUT loads only (K+1) lines of input data for a kernel size of (K×K) and subsequently initiates the calculation of the first convolution layer. The data loading process is regulated by two control signals until the convolution operation is completed. These two control signals (READY and VALID) are utilized to regulate the data flow from DDR to the HePiLUT computation engine. The READY signal denotes the readiness status of the data receiver layer, indicating whether it is prepared to receive data. Conversely, the VALID signal signifies the validity of the data being transmitted by the sender. Within the processing system (PS), the ARM processor has the responsibility of initializing the computation engine on programmable logic (PL) by utilizing the control signal. This process facilitates the dynamic configuration of convolution layer parameters during runtime cycles. Upon the completion of each convolutional operation, the newly calculated data is transferred to the next layer. This data transfer mechanism serves as the loading process for the subsequent CNN layer. The new data supersedes the old data that has already undergone processing. This iterative use of the same memory location multiple times contributes to HePiLUT's enhanced resource efficiency.

The loading process within HePiLUT is visually depicted in Fig. 2. In this representation, considering a convolution kernel size of 5×5 and an input image size of 100×100 pixels. As the kernel size is 5×5, the loading of image data progresses until the 6th line of the input image data (illustrated in steps a, b, and c in Fig. 2). At this point, the calculation process begins alongside with the data loading process. Simultaneously, the calculated values are transferred to the subsequent CNN layer (depicted in steps c, d, e, and f in Fig. 2). In subsequent steps, the loading of input data recommences from the 7th line of the input image data, replacing the previously stored input data in the LUT memory. This iterative process continues until the next layer is filled with data equivalent to the kernel height. If the subsequent layer obtains the necessary data to initiate the calculation process, then all stages of data loading, calculation, and transfer occur simultaneously. This ensures efficient and continuous processing within the HePiLUT accelerator data loading structure.

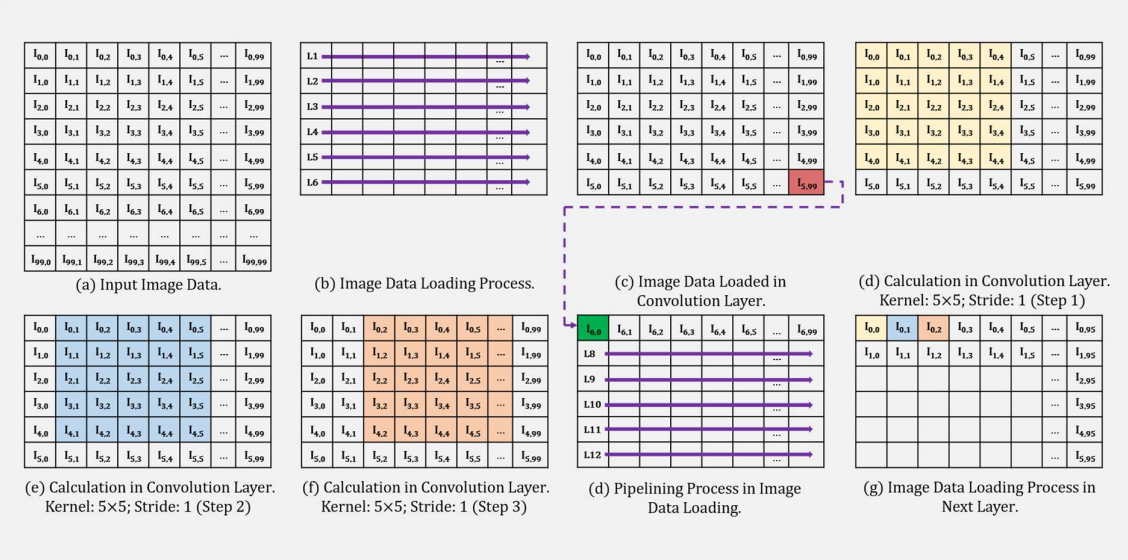

Fig. 2. Pipelining architecture of the proposed HePiLUT CNN accelerator.

2. *Calculation*

In the HePiLUT architecture, the computational engine takes charge of expediting the multiplication and addition functions essential in CNN calculations, employing a pipelined approach for parallelization. Each multiplication and addition array is dedicated to computing the pixels for an individual output feature map. The initiation of the calculation operation begins promptly upon the loading of sufficient input data. For instance, in the scenario where an image comprises a total of 100×100 data and a 5×5 kernel is applied, the convolution calculation operation commences upon receiving the 6th row of input image data. It doesn't wait for the complete loading of all 100×100 data to commence the convolutional calculation operation. Once the convolutional calculation operation concludes, the newly computed data is forwarded to the next layer for subsequent processing in the following clock cycle. This iterative process continues for each CNN layer within the HePiLUT architecture, optimizing the computation by starting calculations as soon as the required input data is available rather than waiting for the complete input data set to be loaded.

3. *Data Transfer*

HePiLUT efficiently manages convolution kernels and feature maps during convolutional operations, creating new feature maps that act as input for subsequent layers. Within a convolution operation, the key processes involve multiplication and addition, both of which demand substantial DSP (Digital Signal Processing) resources. Additionally, the generated feature maps from these operations necessitate considerable memory for buffering. To optimize resource utilization, HePiLUT doesn't store any output feature maps from any CNN layer. Instead, it promptly transfers the calculated values directly into the subsequent layer. This pipelined data transfer approach significantly conserves resources, minimizing the need for storing intermediary feature maps and ensuring a more efficient utilization of memory and processing capabilities.

C. *Design Synthesis*

The HePiLUT CNN accelerator compiler orchestrates convolutional operations utilizing a heterogeneous pipelining architecture, generating a VHDL script for hardware implementation. After completing the design phase and establishing connections among layers, the HePiLUT Intellectual Property (IP) core has been created. To utilize this IP within a block design, it was packaged with an AXI interface, which prevents direct communication with the Processing System (PS) to retrieve image data from DDR. To address this limitation, another IP was specifically designed. This secondary IP receives image data from the PS via Direct Memory Access (DMA) using the AXI stream. It formats the received image data according to the requirements of the HePiLUT IP and then transmits the formatted data to the HePiLUT IP for further processing. The computational resources required on the FPGA are directly proportional to the MAC operation involving input feature maps and kernels for producing output feature maps. Fig. 3 represent the hardware architecture of the HePiLUT. However, the RTL design then synthesis and analysis utilizing synthesis tools such as Xilinx Vivado. This process culminates in the generation of a bitstream file, essential for high-performance FPGA or ASIC implementations.

IV. Result and Discussion

The evaluation process for the HePiLUT CNN accelerator involved both simulation and hardware implementation. The simulation phase was conducted using a designed test bench for the image size of 32×32 pixels, within the Xilinx Vivado environment. An experimental dataset (CIFAR-10) has been constructed to facilitate the simulation. This dataset comprised images with sizes of

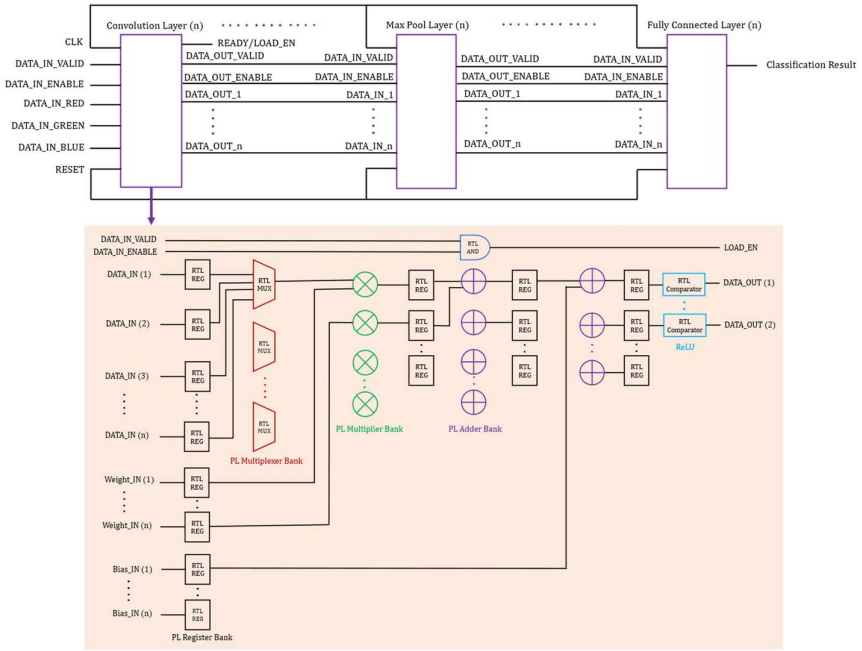

Fig. 3. Hardware architecture of the HePiLUT.

32×32 pixels, with pixel values ranging from 0 to 255. The simulation results were validated by comparing the output sequence with the sequence of input images. The successful match between the result sequence and the input images during simulation provided confidence in the functionality of the HePiLUT accelerator. Following the simulation phase, the HePiLUT accelerator underwent synthesis and was deployed to the Xilinx ZCU102 FPGA board for hardware evaluation. This step involved implementing the accelerator on actual hardware to assess its performance and validate its functionality in a real-world context. The throughput has been measured through the Vivado simulation analysis, resource utilization measured from the Vivado synthesis report, and power measured by the Xilinx Power Estimator (XPE). On the other hand, a run time power measurement also has been conducted to validate the power consumption of the HePiLUT.

A. Data Load Controller Result

HePiLUT serves the purpose of addressing on-chip memory limitations, particularly concerning the deployment of larger networks, all without compromising computational speed. For instance, consider a 32×32 image being processed by HePiLUT. In a conventional convolutional layer, the data loading process would typically occupy a significant amount of memory: 32×32×3, accounting for 3072 bytes. Instead of loading the entire image data, HePiLUT strategically processes data in a more efficient manner. For the same 32×32 image, HePiLUT occupies only 32×4×3, which amounts to 384 bytes. This represents a remarkable reduction, approximately 8x less memory consumption compared to the conventional data loading process.

B. FPGA Implementation Result

The benchmarking results of HePiLUT with VGG-7 CNN model architectures on varied image sizes and datasets showcase its noteworthy performance across several metrics. For image classification on the CIFAR-10 dataset, HePiLUT takes 7716 ns. This translates to an notable classification speed of 129600 FPS at the frequency of 137MHz. In terms of resource utilization, HePiLUT CNN accelerator demonstrates efficiency by utilizing only 4098 LUTs, leveraging its reliance on LUTs for data processing. The power consumption of the HePiLUT is significant, with a consistent consumption of around 3.5W. This underscores the power efficiency of the HePiLUT accelerator. Furthermore, HePiLUT exhibits high accuracy, achieving a classification accuracy of 82.05% for CIFAR-10 image classifications. These findings collectively underscore the efficacy of the HePiLUT accelerator in terms of speed, resource efficiency, and accuracy.

C. Comparison with State-of-the-Art CNN Accelerator

Comparing HePiLUT with other top-notch FPGA-based CNN accelerators reveals its distinct advantages in various performance aspects, such as operating frequency, power consumption, and throughput. Table 1 represents the results of HePiLUT with state-of-the-art CNN accelerator for the 32×32 RGB image classifications and serves as a clear demonstration of HePiLUT's notable efficiency on image classification tasks. The comparative analysis showcases HePiLUT's competitive edge, likely demonstrating the ability of working low (137MHz) frequency, reduced latencies, higher throughputs, or a combination of these factors, illustrating its prominence among FPGA-based CNN accelerators. Its efficient utilization of LUTs for CNN parameter embedding stands out as a distinct advantage, paving the way for enhanced performance metrics compared to its counterparts in the field. This unique approach underscores HePiLUT's resource optimization strategy and its ability to operate efficiently with minimal hardware resources.

In the experimental use case of 32×32 pixel images, FixyFPGA demonstrates higher throughput of 134000 FPS the cost of significantly higher power consumption and higher resource utilization. In contrast, HePiLUT maintaining competitive throughput of 129600 FPS with higher precision. However, HePiLUT demonstrates notable advantages in resource utilization efficiency, with minimal BRAM and DSP utilization compared to the other CNN accelerators for 32×32 RGB image classification. Fig. 4 illustrates the performance evaluation of the proposed HePiLUT CNN accelerator in comparison to FixyFPGA, chosen due to its representation of maximum throughput in contemporary CNN accelerator designs. The comprehensive analysis positions HePiLUT as an competitive solution for CNN acceleration endeavors, characterized by substantial reductions in resource utilization (98% LUT, 99% DSP, and 100% BRAM), alongside energy-efficient operation, manifesting in a 72% reduction in power consumption. However, this enhancement comes with a marginal trade-off, with a 3%mreduction in throughput. Besides, HePiLUT maintains its emphasis on achieving maximum throughput with minimal resource consumption, resulting in a lower operating frequency. This strategic approach ensures that even resource-constrained FPGA devices can effectively leverage the benefits of the HePiLUT accelerator.

TABLE I. EVALUATION OF HEPILUT WITH STATE-OF-THE-ART CNN ACCELEARATOR FOR THE 32×32 RGB IMAGE CLASSIFICATIONS.

Author	Kim et. al [26]	Zhao et. al [27]	FINN [9]	FixyFPGA [22]	HePiLUT
CNN Model	Resnet	BNN	BNN	VGG-7	VGG-7
FPGA	PYNQ-Z1	XC7Z020	ZC706	Stratix 10	ZCU 102
BRAMs	523	94	186	290	0
DSPs	167	3	-	360	1
LUTs	15200	46900	46253	814980	4098
Precision	16-bit	2-bit	1-bit	4-bit	8-bit
Frequency (MHz)	50	143	200	137.29	137
Throughput (FPS)	9.17	168	21906	134000	129600
Power (W)	1.44	4.7	25	22.03	3.45
Efficiency (FPS/W)	6.36	35.74	876	6008	37565

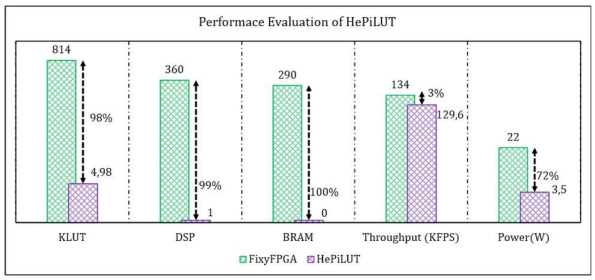

Fig. 4. Performance evaluation of the HePiLUT CNN accelerator.

V. Conclusion and Future Work

The paper introduces HePiLUT, a distinctive LUT-based heterogeneous pipelined CNN accelerator architecture designed for FPGAs. By implementing representative VGG CNN algorithms on the Xilinx ZCU102 FPGA board, the study showcases outstanding performance metrics, reaching a peak performance of 129600 FPS for 32×32 pixel RGB images. Notably, this achievements are accomplished while maintaining minimal resource utilization and low power consumption. This paper outlines potential future avenues for research, including enhancing HePiLUT's versatility and optimizing data and weight transfer efficiency for other state-of-the-art CNN acclerator. Furthermore, the study aims to explore HePiLUT's performance concerning other CNN-based detection and classification models, such as Single Shot Detectors (SSDs). However, this novel pipelining foundation of HePiLUT serves as a stepping stone for further advancements in this domain, promising improvements and innovations in accelerating deep learning models on FPGA platforms.

References

[1] R. A. Amin, M. Hasan, V. Wiese and R. Obermaisser, 'FPGA-Based Real-Time Object Detection and Classification System Using YOLO for Edge Computing,' *IEEE Access*, vol. 12, pp. 73268-73278, 2024.

[2] M. P. Véstias, R. P. Duarte, J. T. de Sousa, and H. C. Neto, 'A fast and scalable architecture to run convolutional neural networks in low density FPGAs', *Microprocessors and Microsystems*, vol. 77, p. 103136, 2020.

[3] Y. Ma, Y. Cao, S. Vrudhula and J. -s. Seo, 'Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA,' *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1354-1367, 2018.

[4] X. Yin, Z. Wu, D. Li, C. Shen and Y. Liu, 'An Efficient Hardware Accelerator for Block Sparse Convolutional Neural Networks on FPGA,' *IEEE Embedded Systems Letters*, Early Access, 2023.

[5] H. -R. Bai, 'A Flexible and Low-Resource CNN Accelerator on FPGA for Edge Computing,' *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, Guangzhou, China, pp. 646-650, 2023.

[6] W. Peisong, and J. Cheng, 'Accelerating convolutional neural networks for mobile applications', *24th ACM international conference on Multimedia*, pp. 541-545, 2016.

[7] X. Wu, Y. Ma, M. Wang and Z. Wang, 'A Flexible and Efficient FPGA Accelerator for Various Large-Scale and Lightweight CNNs,' *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 3, pp. 1185-1198, 2022.

[8] M. P. Véstias et al., 'Fast convolutional neural networks in low density FPGAs using zero-skipping and weight pruning', *Electronics*, vol. 8, no. 11, p. 1321, 2019.

[9] Y. Umuroglu, J. F. Nicholas, G. Giulio, B. Michaela, L. Philip, J. Magnus, and V. Kees, 'Finn: A framework for fast, scalable binarized neural network inference,' *ACM/SIGDA international symposium on field-programmable gate arrays*, pp. 65-74. 2017.

[10] J. Knapheide, B. Stabernack and M. Kuhnke, 'A High Throughput MobileNetV2 FPGA Implementation Based on a Flexible Architecture for Depthwise Separable Convolution,' *30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, pp. 277-283, 2020.

[11] G. Li, F. Li, T. Zhao and J. Cheng, 'Block convolution: Towards memory-efficient inference of large-scale CNNs on FPGA,' *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, pp. 1163-1166, 2018.

[12] A. Ganguly, R. Muralidhar and V. Singh, 'Towards Energy Efficient non-von Neumann Architectures for Deep Learning,' *20th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, pp. 335-342, 2019.

[13] P. R. Sutradhar, M. Connolly, S. Bavikadi, S. M. Pudukotai Dinakarrao, M. A. Indovina and A. Ganguly, 'pPIM: A Programmable Processor-in-Memory Architecture With Precision-Scaling for Deep Learning,' *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 118-121, 2020.

[14] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan and Y. Xie, 'DRISA: A DRAM-based Reconfigurable In-Situ Accelerator,' *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Boston, MA, USA, pp. 288-301, 2017.

[15] M. Gao, G. Ayers, and C. Kozyrakis, 'Practical near-data processing for in-memory analytics frameworks', *International Conference on Parallel Architecture and Compilation (PACT)*, San Francisco, California, USA, pp. 113-124, 2015.

[16] Y. Chen and M. S. Abdelfattah, 'BRAMAC: Compute-in-BRAM Architectures for Multiply-Accumulate on FPGAs', *31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM),* Orlando, FL, USA, pp. 52-62, 2023.

[17] Q. Deng, Y. Zhang, M. Zhang and J. Yang, 'LAcc: Exploiting Lookup Table-based Fast and Accurate Vector Multiplication in DRAM-based CNN Accelerator,' *56th ACM/IEEE Design Automation Conference (DAC)*, Las Vegas, USA, pp. 1-6, 2019.

[18] Y. Umuroglu, Y. Akhauri, N. J. Fraser and M. Blott, 'LogicNets: Co-Designed Neural Networks and Circuits for Extreme-Throughput Applications,' *30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, pp. 291-297, 2020.

[19] E. Wang, J. J. Davis, P. Y. K. Cheung, and G. A. Constantinides, 'LUTNet: Rethinking inference in FPGA soft logic', *27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, San Diego, USA, pp. 26-34, 2019.

[20] B. Zhao, Y. Wang, H. Zhang, J. Zhang, Y. Chen and Y. Yang, '4-bit CNN Quantization Method With Compact LUT-Based Multiplier Implementation on FPGA,' *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1-10, 2023.

[21] Y. Ma et al., 'Scalable and modularized RTL compilation of Convolutional Neural Networks onto FPGA,' *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Lausanne, Switzerland, pp. 1-8, 2016.

[22] J. Meng, S. K. Venkataramanaiah, C. Zhou, P. Hansen, P. Whatmough and J. -s. Seo., 'FixyFPGA: Efficient FPGA Accelerator for Deep Neural Networks with High Element-Wise Sparsity and without External Memory Access,' *31st International Conference on Field-Programmable Logic and Applications (FPL)*, Dresden, Germany, pp. 9-16, 2021.

[23] M. Hall and V. Betz, 'HPIPE: Heterogeneous Layer-Pipelined and Sparse-Aware CNN Inference for FPGAs', *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Seaside, CA, USA, p. 320, 2020.

[24] Y. Zhao et al., 'Automatic Generation of Multi-Precision Multi-Arithmetic CNN Accelerators for FPGAs,' *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, pp. 45-53, 2019.

[25] Y. Fu, E. Wu, A. Sirasao, S. Attia, K. Khan, and R. Wittig, 'Deep learning with int8 optimization on xilinx devices', *White Paper*, 2016.

[26] V. H. Kim and K. K. Choi, 'A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA,' *IEEE Access*, vol. 11, pp. 59438-59445, 2023.

[27] R. Zhao and W. Song, 'Accelerating binarized convolutional neural networks with software-programmable FPGAs,' *ACM International Symposium on Field-Programmable Gate Arrays (ISFPGA)*, pp.15-24, 2017.