High Accuracy and Low Latency Mixed Precision Neural Network Acceleration for TinyML Applications on Resource-Constrained FPGAs

Weisoon Ng[1,2], Wang Ling Goh[1], Yuan Gao[2]
[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[2]Institute of Microelectronics (IME), Agency for Science, Technology and Research (A*STAR), Singapore
Email: WEISOON001@e.ntu.edu.sg, ewlgoh@ntu.edu.sg, gaoy@ime.a-star.edu.sg

***Abstract*—Recent advancements in mixed precision quantization algorithms hold significant promise for Tiny Machine Learning (TinyML) applications that face tight resource constraints. However, to leverage the benefits of mixed precision NNs, specialized hardware is required. The reconfigurability of Field Programmable Gate Arrays (FPGAs) allows tensor computations of any precision. In this work, we propose a streaming architecture with layer-specific processing elements to support interlayer mixed precision NN computations on FPGAs. Additionally, we extend the supported quantization scheme to intralayer level by introducing multiply-accumulation (MAC) units that support intralayer dynamic precision quantization. Bit-serial MAC units are chosen to accommodate more PEs on resource-constrained FPGAs. We successfully implemented a mixed quantized NN for the keyword spotting task on a FPGA-only Digilent A7-35T platform with 62% reduction in resource consumption when compared to the previous FPGA-only implementations. Our implementation also achieves 2 times speedup as compared to the state-of-the-art ASIC implementation while meeting the accuracy requirement of 90% set by MLPerf Tiny Benchmark.**

Keywords— FPGA accelerator, mixed precision quantization, dynamic quantization, multiply-accumulation

I. INTRODUCTION

In recent years, research in mixed precision quantization [1] is quickly gaining attention from both academic and industry due to its ability to maximize Neural Network (NN) compression without compromising the accuracy. This highly efficient network compression technique significantly broadens the landscape of Tiny Machine Learning (TinyML) applications as it allows larger NNs to be implemented on the resource-constrained TinyML devices while maintaining minimal loss in accuracy. Mixed precision quantization scheme can be applied at both interlayer and intralayer levels [1], [2]. Other than mixed precision quantization, dynamic quantization [3] can also be deployed to improve model compression rate. Dynamic quantization preserves the precision of the layers or channels with smaller clipping range without incurring additional bits. Particularly for grouped convolutional NNs with large tensor magnitude difference between channels, intralayer dynamic quantization had demonstrated outstanding network compression ratio over the static quantization [4]. The combined utilization of mixed precision quantization and dynamic precision quantization could present promising opportunity to enhance the efficiency of NN hardware acceleration.

This work was supported by the Agency for Science, Technology and Research (A*STAR), Singapore under the Nanosystems at the Edge Programme, grant no. A18A1b0055.

To fully leverage the benefits of mixed quantized NNs, specialized hardware is necessary. Field Programmable Gate Arrays (FPGAs) have emerged as a compelling choice, offering unparalleled configurability, with ability to support computations with arbitrary precision through their rich reconfigurable Look-Up Table (LUT) resources. However, FPGAs are less explored as TinyML devices compared to microcontroller units (MCUs) and application specific integrated circuits (ASICs) despite having higher performance and higher reconfigurability respectively. This may be due to higher power consumption or higher unit cost for FPGA. To improve FPGA usability in TinyML applications, especially for Internet-of-Things (IoTs) implementations with many edge nodes, it is vital to utilize low-cost and low power FPGAs. However, low-cost FPGAs are often limited by the hardware resources available, leading to performance degradation and unbearable latency. Several studies had also proposed the implementation framework of NNs on low-cost FPGAs as TinyML devices [5], [6]. Many of these studies in fact focused on implementing NNs on System-on-Chip (SoC) FPGAs such as Xilinx Zynq 7020 or Avnet Ultra96 V2, which have on-chip hard processing units, hence incurring higher cost and higher power consumption. It would be beneficial if FPGA-only platforms can be utilized for TinyML applications without compromising the performance and latency.

Hardware architecture selection is also a crucial consideration for TinyML devices. Streaming architectures with cross-layer processing dataflows have been widely studied for their ability to reduce memory footprint and latency [7]. Another advantage of using streaming architectures is that they support layer-specific dataflow and layer-specific PEs, enabling PE customization for efficient interlayer mixed precision NN computation [8]. However, streaming architectures require large number of processing elements (PEs) and more control logics to handle dataflows for different layers as the NNs get deeper. Conversely, architectures with single layer processing dataflows, though incur complex control overhead, require relatively constant control logics, regardless of NN depth. For small NNs with fewer than 10 convolution layers used in TinyML applications, streaming architecture may be preferred over the single layer processing architecture since the complex control overhead in single layer processing architecture will contribute significantly to hardware resource consumption.

By using a streaming architecture and layer-dedicated PEs for mixed precision quantization, the implemented NNs can preserve high accuracy while maintaining very low latency and minimal hardware resource consumption on a

hardware platform. All these advantages enable FPGA to be a more attractive TinyML platform compared to other platforms. This work aims to realize an efficient design for low latency mixed precision NN acceleration for TinyML applications on low-cost resource-constrained FPGAs. The key contributions of this paper are:

- We utilize low resource bit-serial multiply-accumulation (MAC) units in PEs, allowing resource-constrained FPGAs to accommodate more PEs to facilitate the highly efficient streaming architecture at low power.
- We propose the usage of intralayer dynamic precision quantization on top of interlayer mixed precision quantization to improve the network compression rate with minimum extra control overhead.
- We propose a throughput balancing technique that balances the throughput at each layer to maximize the PE utilization rate, further reducing the overall latency.

Although this work focuses on implementing the designs for one of the benchmark applications in Tiny MLPerf [9], namely the keyword spotting (KWS) task, the ideas proposed may also be generalized to other TinyML applications implemented on any FPGA. Through this work, we hope to pave the way for massive adoption of FPGAs in TinyML applications, leveraging the advantages of AI to improve the quality of our daily lives. This paper is organized as follows: Section II discusses the current state of the mixed precision hardware accelerators. The proposed design is explained in detail in Section III, followed by result analysis and comparison with the state-of the-art designs in Section IV. Lastly, a short summary is written to conclude the work in Section V.

II. BACKGROUND AND RELATED WORKS

A. *Mixed Precision Quantization & Architecture*

The management of memory and PEs become challenging when the tensors have varying bit widths for each layer or at each channel. Separate memory cores and computing cores are required for different bit widths, which results in complex control and communication across the computing cores [2]. To address this, BISMO [10] avoids the need for different computing cores by employing a programmable bit-serial computation core that is capable of handling tensors with different bit widths. However, the use of bit-serial computation requires more clock cycles for each MAC operation. Building on the basis of BISMO, N3H [11] presents a heterogenous core where tensors with similar bit widths are computed using the digital signal processing (DSPs), while tensors with irregular bit widths are computed using the bit-serial core. This workload splitting strategy significantly reduces the computation latency induced by bit-serial multiplications while harnessing the advantages of using mixed precision quantization scheme. However, this heterogenous architecture again will incur complex control overhead and dataflow to handle all the different tensors with different bit widths. There are a lot of bit-serial [12, 13] MACs proposed in previous studies, but most of them consume huge resources as their aims is to reduce the latency through different MAC designs.

For TinyML applications, instead of designing a complex controller and computing cores that can handle different tensors with different bit widths from different layers, layer-specific PEs and controllers should be implemented. In this

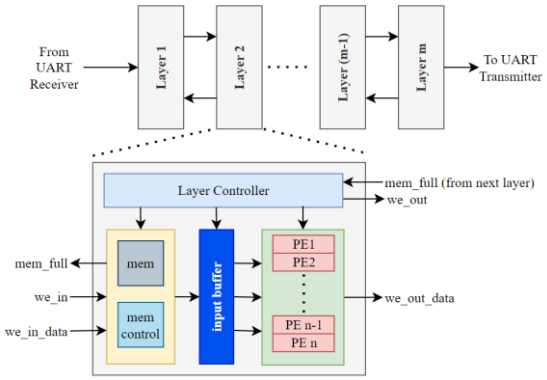

Fig. 1. Proposed streamline architecture.

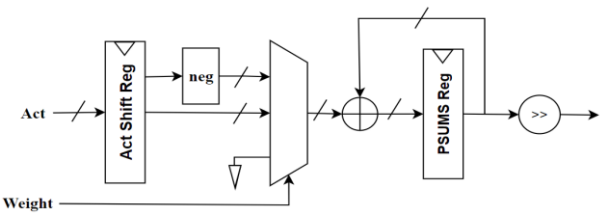

Fig. 2. Low resource bit-serial multiply-accumulation unit.

case, each controller only needs to handle computation of one layer and each PE only needs to compute tensors with fixed bit width, substantially reducing the complexity of the control logics while serving the purpose of handling mixed precision computation. Although this methodology has been proposed in [8, 14], the proposed designed would incur huge number of power-hungry digital signal processors (DSPs) since huge number of PEs are required to facilitate efficient streaming architecture. Besides, they adopt a coarse-grained intralayer mixed precision quantization [8] , which requires tedious NN retraining to preserve the precision of large tensors. In fact, intralayer dynamic precision quantization is a much simpler and direct approach to manage the weights with large distributions within a layer since the large weight distributions often occur across different channels. Furthermore, if streaming architecture with layer-specific PEs is implemented, these PEs can be extended to support intralayer dynamic precision without incurring extra shifter logics.

III. DESIGN AND IMPLEMENTATION

A. *Streaming Architecture*

The proposed streaming architecture consists of distinct hardware blocks catered for each layer in a NN is shown in Fig. 1. The unique characteristic of the streaming architecture allows for customization of hardware modules such as PEs, the layer controller, and the memory management system for each layer. Layer-specific PEs are customized to compute tensors of varying bit widths at different layers, facilitating efficient execution of interlayer mixed quantized NNs. Additionally, the layer controller and memory management system can also be fine-tuned to different dataflows, suiting different type of layers in an artificial NN, such as convolutional layer, depth-wise separable convolutional layer, pooling layer, and fully connected layer. The FPGA's reconfigurability, that permits tailored hardware architectures and dataflows for NNs, presents an opportunity for FPGAs to outperform ASICs and

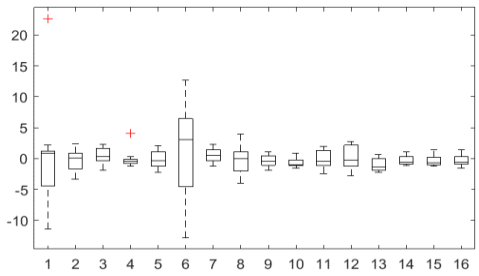

Fig. 3. Channel weight distributions in a grouped convolutional layer.

GPUs, which are primarily designed for general NN acceleration and may lead to suboptimal efficiency in specific NN tasks.

Whenever possible, the output-stationary (OS) dataflow [15] is utilized at each layer so that no additional memory is required to store the PSUMS. To facilitate the OS dataflow, the number of PEs required at each layer must be at least equal to the number of output channel of that layer. Hence, a simple "shift-and-add" bit-serial MAC unit (as shown in Fig. 2) is utilized to minimize the resource consumption of each PE so that more PEs can be accommodated on a resource-constrained FPGA. The rationale behind adopting this MAC design is to reduce the resource consumption but not to address the inherent high latency of bit-serial MACs because this latency induced can be compensated by the efficient streaming architecture. Additionally, the resource allocated for each PE is always optimal because all PEs are customized to support computation of input tensors and PSUMS with different predetermined bit widths at each layer. Considering that the bit widths of weights or activations will directly affect the latency of a layer's computation, the shorter tensors between weights and activations of a layer is selected as the bit-serial input of that layer to reduce the overall latency. Moreover, assuming the ReLU function as the activation function, if activation is the shorter tensor, the negators of the bit-serial MAC units can be omitted to further reduce the resource consumption of the MAC units. A simple zero skipping strategy is also implemented by checking if the input activation is non-positive. If the activation is non-positive, the computation will be skipped, omitting the unnecessary clock cycles.

B. *Interlayer Mixed Precision and Intralayer Dynamic Precision Quantization*

To achieve the minimum bit width without compromising the model accuracy, interlayer quantization is utilized. Different layers in a NN are assigned different bit width based on their relative importance. However, it has been observed that grouped convolutional layers are often assigned larger bit width due to their larger weight distributions (as shown in Fig. 3). This is caused by the batch normalization parameters that have been "folded" [16] into the grouped convolutional layers with larger clipping range to normalize the activations across the separated convolved channels. Consequently, the range of each individual channel varies significantly as shown in Fig. 3.

Although intralayer mixed precision quantization [2] can address this problem of varying weight distributions between channels, it introduces additional control overhead to handle weights with different bit width. Moreover, the search space for the mixed precision quantization will also increase exponentially. A better approach is to dynamically quantize the weights from different channels to different clipping ranges as shown in (1) and (2),

$$S_k = 2 \wedge \{floor\{log_2[max(|W_k|)]\} - B + 2\} \quad (1)$$

$$W_{k,dq} = round\,(min(max(W_k / S_k, -a), a-1)) \times S_k \quad (2)$$

where W_k and $W_{k,dq}$ represents weights of output channel k before and after dynamic quantization, B is the bit width assigned to that layer and a is the clipping range which is equal to $2\wedge(B-1)$. For filters with larger distributions, their precisions are reduced to fit larger number into a similar bit width. As a result, bit width becomes uniform within a layer, easing the design of the control logics. However, it is worth noting that this quantization scheme may lead to larger quantization loss compared to intralayer mixed precision quantization due to reduced precision in certain channels.

To manage the computation of the dynamically quantized weights while aligning the quantization levels of the output activations, a shifter is always needed for the conventional design. However, in the proposed streaming architecture, each PE only handles the computation of one output channel to fulfill OS dataflow. For this reason, the shifter shown in Fig. 2 can be directly replaced by hardwired logics, curbing the need for an additional shifter at each PE to handle the dynamic precision.

C. *Throughput Balancing*

The biggest drawback of the streaming architecture lies in the low PE utilization rate. This shortcoming is due to imbalanced throughput between different layers which causes pipeline to stall. To address this problem, previous studies [14], [17] use unrolling factor, U, and folding factor, F, to balance the throughput of each layer. In this work, adopting bit-serial MACs and zero-skipping strategy provide another degree of freedom to balance the throughput through effective bit, B^*, which can be computed using (3),

$$B^* = \gamma\,(B) + 1 \quad (3)$$

where γ is the probability that the input is greater than zero, B is the bit-width of the bit-serial input, and the term '1' represents the one clock cycle requires to check if the activation is greater than zero. The clock cycles, T, required to compute one pixel of output activation can then be computed as (4),

$$T = (N \times F \times B^*) / U \quad (4)$$

where N is the number of the input activations required to compute one pixel of output activation. Using this methodology, layers which compute faster can have higher bit-width to enhance the accuracy without affecting the overall inference latency.

IV. RESULTS AND DISCUSSION

A. *Experimental Methodology*

In this work, MLPerf Tiny benchmark [9] on keyword spotting (KWS) task is selected as the target application. On-chip measurement procedures for latency, accuracy and energy are therefore strictly adhered to the procedures stated in [9]. A separable convolutional neural network (SCNN) [18] is trained to classify the data. This particular NN is

TABLE II. COMPARISONS WITH STATE-OF-THE-ART IMPLEMENTATIONS ON KWS

NN	SCNN [18]	DSCNN [20]	DSCNN [21]	MLP [19]		SCNN **This work**
Device	XC7A200T (FPGA-only)	GAP9 (ASIC)	NUCLEO-U575ZI-Q (MCU)	XC7A100T (FPGA-only)	XC7Z020 (FPGA-SoC)	XC7A35T (FPGA-only)
Bit Width (W/A)	8/10	-/-	-/-	3/3	3/3	Mixed precision
Model Size (Kb)	1200	-	-	2080	2080	523
LUT BRAM (36Kb)	87K 228	N.A. N.A.	N.A. N.A.	42K 59.5	34K 37	16K 20.5
Frequency (MHz)	47.6	-	-	-	-	100
Top-1 Accuracy (%)	88.1	>90	>90	82.5	82.5	91.01
Average Power (W) Average Latency (ms) Energy/Inference (mJ)	1.04 1.43 1.49	0.056 0.48 0.0267	0.026 38.6 1.0042	1.63* 0.033 0.0537*	1.82* 0.017 0.0309*	0.113/1.20* 0.210 0.0237/0.252*

* The power reported is the power consumed by the whole development board instead of the FPGA core.

TABLE I. SINGLE PRECISION QUANTIZATION VS MIXED PRECISION QUANTIZATION PROPOSED

	FP32	W8/ A12	**MP w dynamic**	MP w/o dynamic
Total size of parameters (kb)	398	100	75.9	75.9
Total size of activation (kb)	1300	487	447	447
Total model size (kb)	1698	587	523	523
Compression ratio	1	2.89	3.25	3.25
Accuracy (%)	92.89	88.9	91.01	69.96

chosen because it is one of the smallest NNs, containing only 12.2K parameters, while achieving high accuracy on the KWS task. The details of the SCNN can be found in [18].

The NN's parameters are quantized using the interlayer mixed precision and intralayer dynamic post-quantization schemes. This is followed by mixed precision quantization on the output activations and PSUMS activation. While this work primarily focuses on the hardware design, we do not aim to achieve the most optimum bit-width assignment. More efficient mixed precision quantization will be left for other works to explore. The mixed precision quantization done in this work is achieved by searching iteratively through the search space to find the optimal bit-width assignment for each layer. The network size and the model accuracy before and after quantization is shown in Table II. The model is quantized using three approaches: 1) quantization of the whole NN to 8-bit weights with 12-bit activations, 2) quantization of the NN using interlayer mixed precision quantization and intralayer dynamic precision quantization scheme, and 3) quantization of the NN using interlayer mixed precision only while maintaining the same compression ratio as 2. The result obtained shows that quantization scheme proposed in this work can maintain high accuracy while achieving a higher model compression ratio.

After the NN is quantized, the NN parameters are exported and implemented on a Digilent Arty A7-35T development board which features a XC7A35T FPGA. The architecture shown in Fig.1 is designed from RTL level. Both synthesis and implementation are done in Xilinx Vivado 2022.1 with minimal manual configuration. This is because this design is meant for implementing NNs on any low-cost FPGAs. Thus, both routing and placement should be automatically done by the software to fit on the targeted FPGA. Table I summarizes the results obtained and compares them with existing state-of-the-arts.

B. Results and Analysis

The effectiveness of our proposed quantization scheme is benchmarked against [18] since both feature the same NN. Reference [19] is a state-of-the-art FPGA implementation of KWS demonstrating excellent energy/inference and latency. Reference [20] and [21] present the state-of-the-art ASIC and MCU implementation respectively on the MLPerf Tiny KWS benchmark. Comparison with these studies provides a holistic view of the effectiveness of the methodology proposed in this work.

Comparing to all the existing work on FPGAs, our work achieves the lowest resource consumption while having the highest accuracy. This result proves the effectiveness of using interlayer mixed precision and intralayer dynamic quantization and low-resource bit-serial MACs. Our implementation achieved 62% reduction in LUT usage compared to the state-of-the-art FPGA-only implementation [19]. However, the achieved latency is lower. This discrepancy in latency is likely attributed to the tradeoff made in [7], where ultra-low precision and higher frequency are prioritized to achieve ultra-low latency at the cost of NN accuracy. Notably, our implementation also achieved 2 times speedup and delivers energy/inference efficiency that is comparable to the state-of-the-art ASIC implementation [20]. Comparing to MCU implementation [21], our FPGA-only implementation outperforms significantly in latency performance. The results prove that FPGA, which offers superior flexibility and adaptability, is an attractive alternative to ASIC and MCU as a TinyML platform.

V. CONCLUSIONS

In conclusion, our work demonstrates the effectiveness of a streaming architecture with layer-specific PE customization, supporting interlayer mixed precision and intralayer dynamic quantization. This implementation achieves a 2 times reduction in inference latency as compared to the state-of-the-art ASIC implementation for the KWS task while maintaining high accuracy. Additionally, we achieve a notable 67% reduction in LUT usage compared to the previous state-of-the-art FPGA-only approach by utilizing bit-serial MAC units. The results emphasize the superior hardware efficiency of the streaming architecture, particularly for TinyML applications with smaller NNs. By using a cross-layer processing streaming architecture and layer-dedicated PEs for mixed precision quantization, FPGAs can achieve low cost, low latency and high flexibility and emerge as a strong player at the TinyML market.

REFERENCES

[1] M. Rakka, M. E. Fouda, P. Khargonekar, and F. Kurdahi, "Mixed-Precision Neural Networks: A Survey," *arXiv preprint*, Aug. 2022, arXiv: 2208.06064.

[2] M. Sun *et al.*, "FILM-QNN: Efficient FPGA Acceleration of Deep Neural Networks with Intra-Layer Mixed-Precision Quantization," *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA)*, 2022.

[3] Z. Zhang, W. Shao, J. Gu, X. Wang, and P. Luo, "Differentiable Dynamic Quantization with Mixed Precision and Adaptive Resolution," *Proc. 38th Int. Conf. Mach. Learning (PMLR)*, Jun. 2021.

[4] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018.

[5] Y. Xu, S. Wang, N. Li, and H. Xiao, "Design and Implementation of an Efficient CNN Accelerator for Low-Cost FPGAs," *IEICE Electronics Express*, vol. 19, no. 19, p. 20220370, 2022.

[6] H.-R. Bai, "A Flexible and Low-Resource CNN Accelerator on FPGA for Edge Computing," *3rd Int. Conf. Neural Netw., Inf. Commun. Eng. (NNICE)*, 2023, pp. 646–650.

[7] H.-J. Kang, "AoCStream: All-on-Chip CNN Accelerator with Stream-Based Line-Buffer Architecture," *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA)*, 2023, pp. 48.

[8] D. T. Nguyen, H. Kim, and H.-J. Lee, "Layer-Specific Optimization for Mixed Data Flow with Mixed Precision in FPGA Design for CNN-Based Object Detectors," *IEEE Trans. Circuits Sys. Video Tech.*, vol. 31, no. 6, pp. 2450–2464, 2021.

[9] C. Banbury et al., "MLPerf Tiny Benchmark," *Proc. Neural Inf. Process. Sys. Track Datasets Benchmarks*, Jun. 2021.

[10] Y. Umuroglu, L. Rasnayake, and M. Sjalander, "BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing," *28th Int. Conf. Field Programmable Logic and Appl. (FPL)*, Jun. 2018.

[11] Y. Gong et al., "N3H-Core: Neuron-designed Neural Network Accelerator via FPGA-based Heterogeneous Computing Cores," *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays (FPGA)*, Dec. 2021, pp. 112–122.

[12] H. Sharma *et al.*, "Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks," in *2018 ACM/IEEE 45th Annual Int. Symp. Comput. Architecture (ISCA)*, Dec. 2017.

[13] J. Albericio *et al.*, "Bit-Pragmatic Deep Neural Network Computing," in *2017 50th Annual IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2017, pp. 382–394.

[14] Y. Zhao *et al.*, "Automatic Generation of Multi-Precision Multi-Arithmetic CNN Accelerators for FPGAs," in *2019 Int. Conf. Field-Programmable Tech. (ICFPT)*, 2019, pp. 45–53.

[15] Z. Du et al., "ShiDianNao: Shifting Vision Processing Closer to the Sensor," *ACM/IEEE 42nd Annual Int. Symp. Comput. Architecture (ISCA)*, 2015, pp. 92–104.

[16] E. Yvinec, A. Dapogny, and K. Bailly, "To Fold or Not to Fold: a Necessary and Sufficient Condition on Batch-Normalization Layers Folding," *arXiv preprint*, Mar. 2022, arXiv: 2203.12646.

[17] Y. Yang, J. S. Emer, and D. Sanchez, "ISOSceles: Accelerating Sparse CNNs through Inter-Layer Pipelining," in *2023 IEEE Int. Symp. High-Perform. Comput. Architecture. (HPCA)*, 2023, pp. 598–610. doi: 10.1109/HPCA56546.2023.10071080.

[18] G. Dinelli, G. Meoni, E. Rapuano, G. Benelli, and L. Fanucci, "An FPGA-Based Hardware Accelerator for CNNs Using On-Chip Memories Only: Design and Benchmarking with Intel Movidius Neural Compute Stick," *Int. J. Reconfig. Computing*, Oct. 2019.

[19] H. Borras et al., "Open-source FPGA-ML codesign for the MLPerf Tiny Benchmark," *Proc. Neural Inf. Process. Syst. Track Datasets Benchmarks*, Jun. 2022

[20] GreenWaves Tech., "GAP9 Achieves the Lowest Energy Consumption on All MLPerf Tiny v1.0 Benchmarks," https://greenwaves-technologies.com/lowest-energy-consumption-on-all-mlperf-tiny-v1-0-benchmarks *(accessed Jul. 24, 2022*).

[21] MLCommons, "Inference: Tiny v1.1 Results." https://mlcommons.org/en/inference-tiny-11 (*accessed Jun 27,2023*).