FPGA Acceleration for Scalable High-Resolution OPIR Target Detection

Daniel C. Stumpp, Alan D. George
ECE, University of Pittsburgh
NSF SHREC Center
Pittsburgh, PA, USA
{daniel.stumpp,alan.george}@pitt.edu

***Abstract*—The task of infrared small-object segmentation has a wide range of applications. One such application is in military early-warning systems where overhead persistent infrared (OPIR) sensors are leveraged for target detection. Although targets of interest often manifest as dim point-source targets, making them difficult to detect, recent machine-learning algorithms have enabled significant advances in detection capability. However, these more complex algorithms and increasing sensor resolution have made high-throughput, on-orbit processing challenging. This research explores FPGA acceleration of the Multiscale Local Contrast Learning Network (MLCLNet) target-detection model. MLCLNet was selected for its combination of high detection performance and architectural simplicity. Effects of model quantization required for acceleration were evaluated and shown to be minimal and, in some cases, positive. Additionally, the Xilinx Deep Learning Processor Unit (DPU) performance for the inference task is evaluated on the Xilinx UltraScale+ and Versal AI Core device architectures. Six DPU configurations and six MLCLNet model sizes were used to parameterize inference with 128×128 subframes. This research demonstrates that the Versal DPU can perform subframe inference at up to 1626 FPS with up to 6.7× speedup over the older UltraScale+ architecture. Informed by these findings, the Versal architecture's heterogeneous nature is leveraged to implement an accelerated end-to-end target-detection pipeline. This pipeline enables inference on large OPIR frames by batching them into appropriately overlapped subframes, preprocessing, performing inference, and postprocessing into detections. Throughput ranging from 0.96 FPS to 75.79 FPS is achieved for frame sizes ranging from 4k×4k down to 500×500. The proposed architecture's resource utilization, latency, and power consumption are also analyzed.**

***Index Terms*—FPGA acceleration, machine learning, overhead persistent infrared, remote sensing, target detection, Versal**

I. Introduction

Missile early-warning systems rely on space-based overhead persistent infrared (OPIR) sensors to detect and monitor for potential threats [1]. These threats often appear as small point-source targets with low signal-to-noise (SNR) ratios. The combination of low SNR, small size, and complex backgrounds make these targets challenging to detect. Increasingly sophisticated algorithms have been developed in response to these challenges, with machine learning (ML) being a dominant area of algorithm advancement [2]. Due to the time-sensitive nature of the target-detection task and potential limitations to downlink bandwidth, the ability to perform low-latency on-orbit computation is highly desirable. Successfully enabling the on-orbit deployment of ML-based detection networks requires embedded acceleration. Additionally, optimizations to reduce compute complexity, such as quantization, cannot adversely impact detection performance.

Existing evaluations of ML methods have identified various models capable of a high probability of detection for OPIR targets. In this research, we select the Multiscale Local Contrast Learning Network (MLCLNet) architecture for acceleration due to its comparatively small size and computational complexity, architecture simplicity, and overall detection capability [3]. Various size MLCLNet models are trained and evaluated using a custom multi-target OPIR dataset. The Xilinx Vitis and Vitis-AI development tools are used for model quantization, deployment, and hardware development. After analyzing the effects of quantization on MLCLNet, we evaluate the capability of the Xilinx Deep Learning Processor Unit (DPU) for inference acceleration. This evaluation directly compares the Xilinx Versal Adaptive Compute Acceleration Platform (ACAP) and the older UltraScale+ family of Xilinx devices. After benchmarking the model configurations and devices of interest, the results inform the development of an end-to-end pipeline designed to perform preprocessing, inference, and postprocessing. This proposed architecture leverages the heterogeneous nature of the Versal device to enable high-throughput OPIR target detection. The pipeline is developed and evaluated for arbitrarily large input frames by processing inputs using a batched subframe approach. This research, therefore, serves as both a presentation of an application-specific acceleration architecture and a case study in the acceleration of ML for high-resolution remote-sensing imagery using the Xilinx Versal ACAP.

II. Related Research

There has been significant development of single-frame infrared target detection methods in recent years. However, little focus has been paid to their complexity and performance on embedded platforms. Most included complexity evaluations use desktop- or datacenter-grade devices; however, there are some exceptions for older methods. The research in [4] focuses on GPU acceleration of the IPI (Infrared Patch Image) method for single-frame infrared target detection. They achieve a 20× speedup over the CPU implementation when using an NVIDIA Jetson AGX Xavier. In [5] an FPGA implementation of a filter-based infrared target-detection method is

introduced. This simple algorithm can easily achieve real-time performance. However, the detection performance is much worse than the current state-of-the-art. Another similar design implements a TopHat-filtering approach on an FPGA [6]. The implemented TopHat design reduces computation time by 25.9% compared to a CPU implementation. GPU- and FPGA-accelerated background-suppression algorithms for infrared target detection were introduced in [7] and [8], respectively. However, they used desktop-grade devices and the current state-of-the-art has far exceeded their detection performance.

The research in [9] explores the acceleration of a convolutional layer for infrared target detection. However, the network is simple and not representative of the challenges faced when accelerating existing state-of-the-art models. Although no research has been found where full ML methods specific to infrared target detection were accelerated on embedded platforms, there is research on more general ML acceleration. For instance, the research in [10] leverages Xilinx Vitis-AI to accelerate generic object-detection models, such as YOLOv3. However, it cannot provide the required insight into the acceleration of application-specific networks such as MLCLNet that formulate the task as a small-object segmentation problem. Such previous research also lacks insight into the need for an end-to-end solution for embedded processing of large input frames.

III. Background

This section provides a brief background relevant to this research. First, the task of OPIR target detection and existing methods are discussed. A more detailed discussion of MLCLNet and the Xilinx Versal ACAP then follows.

A. OPIR Target Detection

The dynamic and unpredictable nature of OPIR scene backgrounds has led to the increasing use of single-frame detection methods. A variety of different algorithms including filter-based methods [11]–[14], local contrast methods [15]–[20], and ML methods [3], [21]–[26], among others, have been proposed for the task of infrared small-object segmentation. The evaluations in [27] demonstrated that many of these ML methods can be successfully trained for the specific task of OPIR target detection. The existing methods were evaluated for this research and MLCLNet [3] was chosen due to its small size, detection performance, and Xilinx DPU compatibility.

B. Multiscale Local Contrast Learning Network

The MLCLNet architecture for small target detection is shown in Fig. 1. The encoder-decoder structure of the model is built upon ResNet-20 [28] for feature extraction and FPN [29] for feature fusion. Multiscale Local Contrast Learning (MLCL) modules reside on the skip connections and help extract dim, small targets of interest. The model output is a binary segmentation mask that indicates the predicted label of each pixel in the frame—either target or background. When performing the proposed experiments, the model architecture remains the same but the number of channels in each of the three main ResNet-20 stages, denoted by x_1, x_2, and x_3, is varied to evaluate the effects of model size.

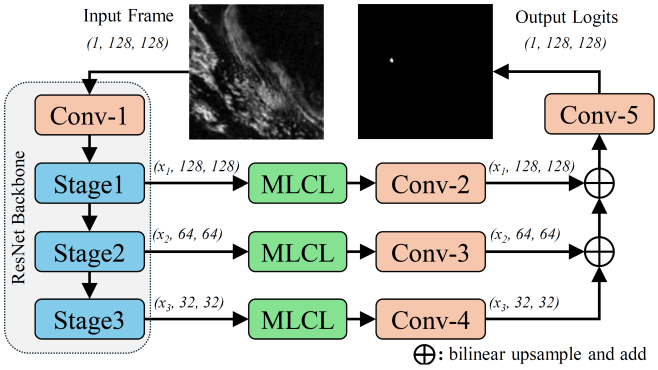

Fig. 1. MLCLNet architecture derived from [3] with layer output sizes for 128×128 subframe input.

The MLCL layers of the network enable the segmentation of small, often unresolved, targets that traditional methods struggle with. The MLCL layer is inspired by traditional local-contrast detection methods. The ALCNet architecture [22] was the first to introduce a local-contrast prior. However, MLCLNet takes the prior one step further by architecting the model to learn local contrast using neural network layers. Specifically, the convolutional and dilated-convolutional layers are used. Using dilated convolution within the MLCL layers allows the local contrast to be computed across multiple scales.

C. Xilinx Versal ACAP

The primary device of interest in this research is the AMD Xilinx Versal ACAP. Specifically, the VCK190 AI Core development board because it contains dedicated hardware for accelerating ML workloads [30]. The forthcoming AI Edge series also includes dedicated ML hardware and is designed for low-power systems. The AI Edge is discussed when considering the results of this research but was not yet available for direct comparison. The Versal AI Core is a highly heterogeneous architecture including an application processing unit (APU), FPGA programmable logic (PL), AI Engine (AIE) vector processors, and a programmable network on chip (NoC) [31]. The APU is a dual-core Arm Cortex-A72 and is accompanied by an Arm Cortex-R5F real-time processing unit. The PL includes look-up tables (LUT), flip-flops (FF), block RAM (BRAM), ultra RAM (URAM), and digital signal processors (DSPs). The AIEs are high-performance vector processors implemented in a 2D grid. These device regions and off-chip DDR memory are tied together using the programmable NoC [31]. While the AI Edge series is specifically targeted for edge applications, the AI Core also has edge applicability and has been explored for such applications [32], [33]. Therefore, the Versal AI Core VCK190 is considered an appropriate target as an embedded processing device in the context of this analysis.

IV. Methods

The following sections present an overview of model configuration, training, and quantization. Methods used for the

TABLE I
SUMMARY OF TRAINED MODEL CONFIGURATIONS FOR EVALUATION.

Metric	Model Configuration					
	c2	**c4**	**c8**	**c16**	**c32**	**c64**
Params (M)	0.01	0.03	0.14	0.55	2.20	8.78
GOP	0.06	0.23	0.89	3.52	13.99	55.84
nIoU	0.51	0.67	0.69	0.70	0.70	0.69
AUC	0.88	0.93	0.93	0.94	0.93	0.94
Pd	0.69	0.82	0.83	0.84	0.84	0.85
Fa	0.02	0.10	0.11	0.22	0.18	0.38

DPU benchmarks are then discussed. Finally, the end-to-end acceleration architecture is introduced.

A. Dataset, Training, and Quantization

A variety of different MLCLNet model configurations are evaluated in this research. The number of filters in each backbone layer is adjusted to scale the model. The number of filters used for a given model is denoted $[x_1, x_2, x_3]$. The values are constrained so that $x_2 = 2x_1$ and $x_3 = 2x_2$ as is the case in [3]. Model configurations are denoted by the shorthand notation $\mathrm{c}(x_1)$ and configurations c2, c4, c8, c16, c32, and c64 are evaluated, where c16 is the default from [3].

The models are trained on a custom multi-target OPIR dataset generated using the Air Force Institute of Technology Sensor and Scene Emulation Tool (ASSET) [34], [35]. This dataset generation follows the steps we introduced in [27], except up to 10 targets are injected per frame. The frame size is 128×128. Each model is trained for 20 epochs with early stopping using a validation set to select the best model weights. Table I reports model complexity and target-detection performance for each model configuration before quantization. Complexity is reported in terms of millions of parameters and giga-operations (GOP) for inference. Four metrics are used to quantify detection performance. Normalized intersection over union (nIoU) and area under the curve (AUC) are pixel-level metrics evaluating segmentation performance. Probability of detection (Pd) and false alarm rate (Fa) are target-level metrics specific to the target-detection task. Detectors with Pd approaching one and Fa approaching zero are best. MLCLNet outperformed other target detection methods evaluated in [27] (e.g. DNANet [24] and ALCNet [22]), having equivalent or better performance in all cases. This performance along with the network's relatively low computational complexity motivated the choice of MLCLNet for acceleration.

Vitis-AI 3.0 is used to quantize the PyTorch floating-point model into an 8-bit integer (INT8) representation [36]. The quantization is performed using power-of-two scaling, the only method supported on the DPUs due to the significant performance benefits compared to floating-point scaling. Calibration is performed using the *MinMax* method and 1,000 input frames sampled from the validation set.

B. DPU Benchmark Configuration

A subset of available Xilinx DPU configurations are evaluated for each device family used [36]. The ZCU104 [37] and ZCU102 [38] development boards are used as UltraScale+ reference devices. For the UltraScale+ DPU (DPUCDX8G), we only evaluate the most performant configuration available for each device (B4096x2 for ZCU104 and B4096x3 for ZCU102). These DPU configurations include the largest convolutional engine with two and three compute units (CUs), respectively. We evaluate four different Versal AI Core DPU (DPUCVDX8G) configurations for the VCK190. Configurations are denoted as C#B#, where the number following C indicates the number of AIE tiles used for each batch. The number following B indicates the batch size used. Both single-batch configurations and max-batch-size configurations are evaluated. The configurations evaluated are C32B1, C64B1, C32B6, and C64B5. All other parameters are fixed in their most performant configuration. Multiple CU variants of the Versal DPU are not evaluated because they do not enable batch processing as preferred for subframe inference. Performance is measured using the Vitis-AI Runtime (VART) `xdputil benchmark` utility. The DPU is built and deployed using Vitis 2022.2. The PL is clocked at 333 MHz and the AIEs use a 1.25 GHz clock.

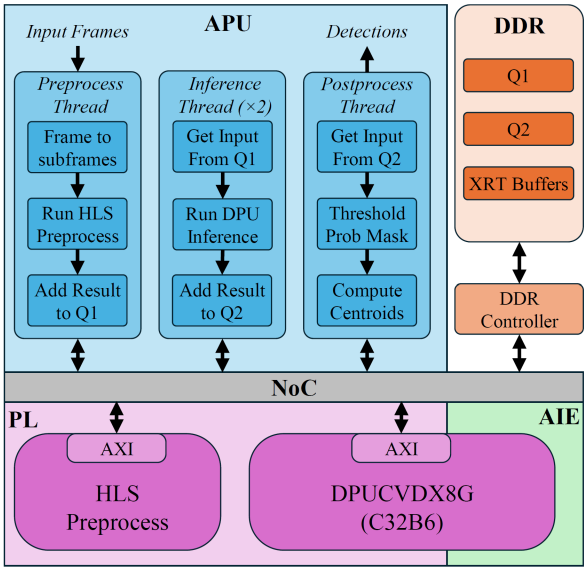

Fig. 2. End-to-end acceleration architecture targeting Versal AI Core.

C. End-to-End Acceleration Architecture

The DPU benchmarking results are used to inform architecture decisions for an end-to-end target-detection pipeline. In addition to model inference, this pipeline must include required preprocessing and postprocessing. Because the primary application constraint is the need for high throughput and low latency, preference is given to the most performant DPU configuration. As shown in Section V-B, this is the C32B6 DPU on Versal AI Core. The proposed end-to-end architecture is built around this DPU and the heterogeneous Versal AI Core architecture as shown in Fig. 2. The architecture leverages high-level pipelining at the subframe level with multiple threads simultaneously running on the APU.

The PL is utilized for the preprocessing kernel and DPU and the AIEs are used for the DPU. Each of these domains and DDR memory are connected via the NoC. The pipeline enables efficient processing of raw input frames to point detections.

The preprocessing thread is responsible for reading the raw 16-bit OPIR frames, converting each into a stack of 128×128 subframes, performing normalization, and then adding the preprocessed subframes to the inference queue (Q1). Conversion of large frames into a stack of subframes allows for batched inference and processing of large remote-sensing imagery. To prevent missed detections when the target falls on the edge of a subframe, the subframes are overlapped by δ on each side. Therefore, the total number of $S \times S$ subframes for a square input frame size $F \times F$ is defined as $\lceil (F-S)/(S-\delta)+1 \rceil^2$. The value of δ is set to three for the included experiments. The input frames are split into batched subframes using pre-computed regions. The subframes are then passed to a preprocessing PL kernel implemented with Vitis high-level synthesis (HLS). The preprocessing kernel subtracts the normalization mean from each element, multiplies by the inverse of the normalization standard deviation and applies the DPU gain determined during the quantization process to produce a normalized INT8 value for input to the DPU. The kernel processes 32 elements simultaneously using internal parallel streams and operates with a 333 MHz clock frequency. Xilinx Runtime (XRT) is used to move data to and from the preprocessing kernel using an AXI master interface.

The inference thread takes the preprocessed data from Q1 and launches inference using the VART DPU runner. When the inference is complete, it adds the results to the postprocessing queue (Q2) and moves on to the next batch. Two instances of the inference thread are used to keep the DPU processing pipeline full. In the final stage, the postprocessing thread takes inference results from Q2 and thresholds the raw logit outputs to create a binary segmentation mask for the associated input subframe. OpenCV is then used to find the centroid of each segmented detection cluster. These centroid coordinates are then converted into the corresponding original coordinates of the $F \times F$ frame and written as an output detection. The software is deployed on the APU running at a clock frequency of 1.2 GHz and is compiled using "-O3" compiler optimization. The base device platform is stripped of all unnecessary IP to ensure accurate resource and power results.

V. Experiments and Results

Experimentation focused on three main areas: quantization effects analysis, MLCLNet benchmarking on DPUs, and characterization of the proposed end-to-end acceleration architecture. All configuration and deployment parameters are as specified in Section IV unless otherwise stated.

A. Quantization Effects

Model quantization was performed as described in Section IV-A and the effects on model detection performance were evaluated. Performance metrics were re-computed using the INT8 quantized model. The effect of quantization on these detection performance metrics is shown in Fig. 3. Apart from Fa, all metrics change less than 6.0% from the floating-point baselines. However, there is a substantial drop in false alarms for all models except c2. Model configurations c16, c32, and c64 all see a >50% reduction in Fa. This result points to a regularization effect of the quantization that improves the operating point of the detector by reversing the effects of overfitting that were introduced to the larger models during training. Overall, the negative effects of quantization are limited, and in some cases, the positive effect of reduced Fa is observed. Based on this result, using INT8 quantization to leverage the reduced computational complexity and 4.0× reduction in parameter memory is justified for the MLCLNet architecture when applied to on-orbit OPIR target detection.

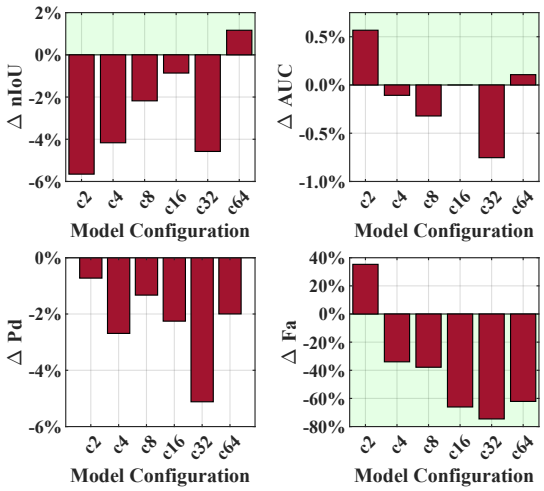

Fig. 3. Effects of quantization on detection performance metrics reported as a percentage change from original 32-bit floating-point model detection performance. Shaded green indicates a positive impact on detection capability.

B. DPU Benchmark Results

The results of the DPU benchmark evaluation are shown in Table II. Both 128×128-subframe FPS and accelerator efficiency are presented. Accelerator efficiency is computed by dividing the number of giga-operations per second (GOPS) by the peak theoretical GOPS of the DPU configuration. For all DPU configurations, the FPS decreases, and efficiency increases as the model size increases. Efficiency is generally higher for the UltraScale+ DPU configurations due to the lower theoretical GOPS. Overall the small model size needed for this task keeps efficiency low, particularly for the Versal DPU configurations, which achieve a max accelerator efficiency of 39.4%. The size of the MLCLNet layers and the DPU architecture drive these inefficiencies. For example, the DPUCVDX8G implements 16-channel parallelization, meaning convolutional layers with less than 16 channels cannot fully utilize the available hardware.

Despite the observed DPU inefficiencies, the FPS performance is strong, particularly for the Versal DPU. The performance ranges from 317 FPS for c64 to 1626 FPS for c2. The max-batch-size configurations of the Versal DPU outperform

TABLE II
SUBFRAME FPS (AND ACCELERATOR EFFICIENCY) FOR EACH DPU AND MODEL COMBINATION. BEST RESULTS IN BOLD.

Device (DPU)		ZCU104 (DPUCZDX8G)	ZCU102 (DPUCZDX8G)	VCK190 (DPUCVDX8G)			
DPU Configuration		B4096x2	B4096x3	C32B1	C64B1	C32B6	C64B5
Model Config	c2	402 (0.9%)	538 (0.8%)	413 (0.2%)	396 (0.1%)	**1626** (0.2%)	1461 (0.1%)
	c4	391 (3.3%)	509 (2.8%)	409 (0.9%)	392 (0.4%)	**1484** (0.5%)	1361 (0.3%)
	c8	343 (11.3%)	422 (9.1%)	389 (3.4%)	368 (1.6%)	**1362** (2.0%)	1261 (1.1%)
	c16	242 (31.5%)	296 (25.4%)	313 (10.7%)	304 (5.2%)	**1033** (5.9%)	994 (3.4%)
	c32	109 (56.3%)	141 (48.1%)	192 (26.2%)	195 (13.3%)	**621** (14.1%)	619 (8.5%)
	c64	35 (72.8%)	48 (64.8%)	72 (39.4%)	97 (26.3%)	258 (23.4%)	**317** (17.3%)

their unbatched counterparts by up to 3.9× and 3.7× for the C32B6 and C64B5 configurations, respectively. Comparing the C32B6 and C64B5 configurations directly shows the C32B6 configuration outperforming on all but the c64 model, despite the C64B5 configuration having 1.67× higher peak theoretical performance due to its use of more AIE tiles. Again, the small model size contributes to this result as only the largest model can sufficiently utilize the additional AIE tiles to compensate for having one less frame per batch. These results are promising for performance on the forthcoming Versal AI Edge series of devices, as the max-batch-size DPU for those devices uses a C20B14 configuration with specialized AIE-ML tiles [39]. The focus on larger batch-size ML processing will benefit inference for these smaller models and batched subframe processing of large remote-sensing imagery.

Compared to the UltraScale+ architecture, the Versal AI Core outperforms by up to 6.7×. This advantage is particularly significant with increasing model size. Using AIEs as the main computational leaves additional resources in the PL for other kernels, such as the preprocessing kernel proposed, and has been previously found to enable more energy-efficient inference [33]. Based on the benchmark results, the C32B6 DPU configuration is selected for use in the end-to-end acceleration architecture.

C. End-to-End Architecture Performance

The end-to-end acceleration architecture is evaluated based on resource utilization, throughput, latency, and power consumption. Input frame sizes ranging from 500×500 to 4k×4k are used. Only model configurations c4, c8, and c16 are evaluated based on the target-detection results from Table I and analysis of quantization effects. The c2 configuration does not achieve satisfactory detection performance and the marginal benefits of models larger than c16 are reduced by quantization and do not justify the reduction in throughput performance.

1) Resource Utilization: Fig. 4 shows the resource utilization for the end-to-end architecture, split between the platform, preprocessing kernel, and the DPU as a percentage of the total available resources on the VCK190. The DPU dominates resource utilization, with all other contributions nearly negligible except the preprocessing kernel's use of 96 DSPs. All resources are <50% utilized except for BRAM and URAM, which are heavily utilized by the DPU's batch handler. AIE utilization by the DPU remains under 50% due to the use of C32B6 instead of C64B5.

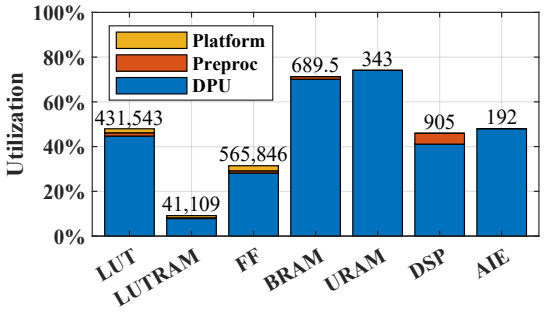

Fig. 4. Resource utilization of end-to-end pipeline as a percentage of available resources. Absolute utilization is reported at the top of each bar.

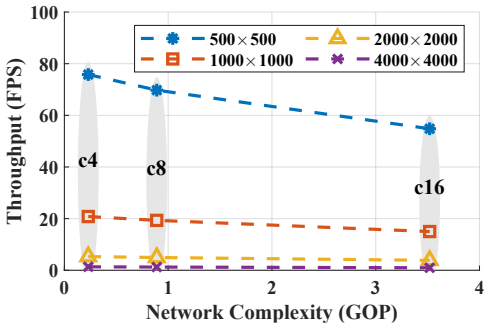

Fig. 5. End-to-end throughput vs. complexity for selected frame sizes.

2) Throughput: Throughput was measured for an input stack of 100 OPIR frames of each selected size. The results are shown in Fig. 5 with respect to the model network complexity in GOP. FPS performance ranges from 0.96 for c16 with 4k×4k input frames to 75.79 for c4 with 500×500 input frames. For this range of network complexity, the relationship with throughput is roughly linear, however, if larger model sizes were to be used, the rate of decrease in throughput would slow as the accelerator efficiency increases. The end-to-end architecture is pipelined, meaning that most non-inference processing should be hidden and the throughput should approach that inferred from the subframe inference benchmark. A 4k×4k input frame, for example, is split into 1024 subframes for processing meaning the theoretical max inference of the end-to-end pipeline is 1 FPS based on the benchmarking results in Table II. The end-to-end pipeline achieves 96% of this theoretical max, indicating successful

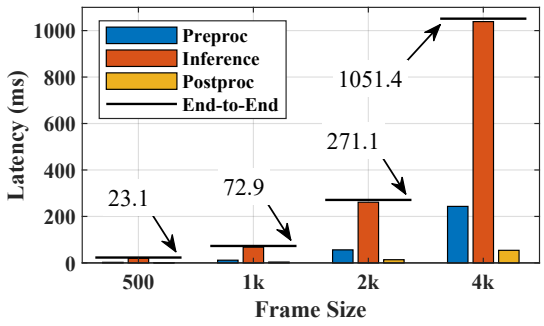

Fig. 6. Single-frame latency for c16 model configuration across various frame sizes. Annotated lines indicate the total end-to-end latency for each frame size.

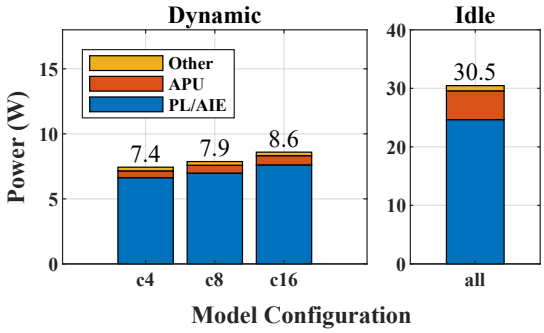

Fig. 7. Dynamic (left) and idle (right) power for end-to-end acceleration architecture. Idle power is independent of the model configuration used.

pipelining of computation. The following unbatched latency results further confirm this conclusion.

3) Latency: The latency of each stage in the pipeline and the total end-to-end latency are reported in Fig. 6 for the c16 model configuration run on each input frame size. The inference latency is the dominant component for every frame size, taking 4.3× to 8.0× longer than the preprocessing latency for the 4k×4k and 500×500 frame sizes, respectively. In all cases, the end-to-end latency is less than the sum of individual stage latencies enabled by the pipelined architecture. The overhead beyond the inference latency is as low as 1.2% for the 4k input frames. The latency of the postprocessing stage can vary depending on the number of detections, but it is consistently far below the other contributing latencies. Additionally, because the architecture is pipelined at the subframe level, the latency for detections in any given subframe may be considerably lower. By selectively ordering subframes based on past detections or predefined regions of interest, the effective latency to detection for the most critical targets can be reduced to a small fraction of the reported results.

4) Power: Power consumption of the architecture on the VCK190 was measured using the Board Evaluation and Management (BEAM) tool provided by Xilinx [40]. The results are reported in Fig. 7. Idle power is collected without running tasks and is independent of model configuration. The average dynamic power is measured for each model configuration and frame size combination. There is no dependence on the size of the input frame, therefore the average of all frame sizes is provided for each model configuration. Idle power is the dominant draw for the architecture, constituting >80% of the total power. The PL and AIE, where the resource-intensive DPU is located, constitute the vast majority of idle and dynamic power for all configurations, followed by the APU and some additional power consumption by other device components. Accounting for the FPS in Fig. 5, the accelerator achieves an FPS/Watt performance ranging from 2.0 for c4 with a frame size of 500×500 to 0.02 for c16 and a 4k×4k input frame. Normalizing the results by the number of subframes, we observe 25.3, 33.2, and 36.1 subframe FPS/Watt for the c4, c8, and c16 configurations, respectively. With the Versal AI Edge being designed with power efficiency in mind and the anticipated benefits of its corresponding DPU, it may yield further improvements in throughput-per-Watt.

VI. Conclusion

This research has leveraged the Xilinx Versal AI Core device architecture to accelerate ML-based OPIR target detection. The MLCLNet model was selected and six model configurations were trained and evaluated using a custom dataset. The effects of model quantization on Pd were minimal and quantization reduced the Fa for most model configurations. All model configurations were benchmarked using six DPU configurations across three test devices. The Versal AI Core was shown to outperform the UltraScale+ devices by up to 6.7× and achieved up to 1626 FPS for 128×128 subframes. The higher batch-size configurations outperformed unbatched configurations by up to 3.9×, suggesting possible future benefits from the AI Edge DPU tailored for larger batch sizes.

Benchmark results motivated the selection of the C32B6 DPU configuration for the end-to-end acceleration architecture. The architecture enables arbitrary-sized input frames to be processed using batched inference on appropriately overlapped subframes. The model preprocessing is accelerated using a PL accelerator developed with HLS. The architecture is pipelined at the subframe level by running multiple simultaneous threads on the APU, enabling the majority of preprocessing and postprocessing latency to be hidden during model inference. The end-to-end pipeline was evaluated on OPIR input frames ranging from 500×500 to 4k×4k in size. Performance ranged from 0.96 to 75.79 FPS for the various model and size combinations. Latency for the most complex model configuration (c16) ranged from 23.1 ms to 1.10 sec. However, the latency-to-detection for regions of interest can be significantly reduced by selectively prioritizing the corresponding subframes. Although the size and required processing rate of remote-sensing systems vary and are often unavailable (in the case of OPIR), the provided results effectively characterize current capability and are within the range of expected requirements. The architecture presented demonstrates a case study in embedded, high-throughput, low-latency ML processing of high-resolution remote-sensing imagery leveraging the Versal AI Core device. This research demonstrates up the possibility of on-orbit processing for the OPIR target-detection task and others like it.

ACKNOWLEDGMENTS

This research was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783. The authors thank the MITRE Corporation for assistance with dataset generation.

REFERENCES

[1] M. Zhao, W. Li, L. Li, J. Hu, P. Ma, and R. Tao, "Single-frame infrared small-target detection: A survey," *IEEE Geoscience and Remote Sensing Magazine*, 2022.

[2] R. Kou, C. Wang, Z. Peng, Z. Zhao, Y. Chen, J. Han, F. Huang, Y. Yu, and Q. Fu, "Infrared small target segmentation networks: A survey," *Pattern Recognition*, vol. 143, p. 109788, 2023.

[3] C. Yu, Y. Liu, S. Wu, Z. Hu, X. Xia, D. Lan, and X. Liu, "Infrared small target detection based on multiscale local contrast learning networks," *Infrared Physics & Technology*, vol. 123, p. 104107, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1350449522000883

[4] X. Hao, Y. Liu, F. Song, T. Lei, Y. Cui, and Y. Zhang, "Gpu-accelerated infrared patch-image model for small target detection," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2022, pp. 179–188.

[5] Y. Wu, H. Yan, and M. Wang, "Detection algorithm of infrared dim small target based on fpga," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 7650–7655.

[6] Y. Zhang, J. Gao, X. Yang, and C. Yang, "Hardware acceleration of infrared small target detection based on fpga," in *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2022, pp. 342–347.

[7] X. Wu, J.-q. Zhang, X. Huang, and D.-l. Liu, "Separable convolution template (sct) background prediction accelerated by cuda for infrared small target detection," *Infrared Physics & Technology*, vol. 60, pp. 300–305, 2013.

[8] Z. Wang, H. Song, H. Xiao, W. He, J. Gu, and K. Yuan, "A real-time small moving object detection system based on infrared image," in *2014 IEEE International Conference on Mechatronics and Automation*. IEEE, 2014, pp. 1149–1154.

[9] R.-h. Cai, L.-q. Song, P. Li, and B.-t. Zhang, "Acceleration of convolution layer in fpga of infrared target detection algorithm," in *AOPC 2020: Infrared Device and Infrared Technology*, vol. 11563. SPIE, 2020, pp. 146–151.

[10] J. Wang and S. Gu, "Fpga implementation of object detection accelerator based on vitis-ai," in *2021 11th International Conference on Information Science and Technology (ICIST)*. IEEE, 2021, pp. 571–577.

[11] S. D. Deshpande, M. H. Er, R. Venkateswarlu, and P. Chan, "Max-mean and max-median filters for detection of small targets," in *Signal and Data Processing of Small Targets 1999*, vol. 3809. SPIE, 1999, pp. 74–83.

[12] M. Zeng, J. Li, and Z. Peng, "The design of top-hat morphological filter and application to infrared target detection," *Infrared physics & technology*, vol. 48, no. 1, pp. 67–76, 2006.

[13] P. Wang, J. Tian, and C. Q. Gao, "Infrared small target detection using directional highpass filters based on ls-svm," *Electronics letters*, vol. 45, no. 3, pp. 156–158, 2009.

[14] S. Kim, "Min-local-log filter for detecting small targets in cluttered background," *Electronics letters*, vol. 47, no. 2, p. 1, 2011.

[15] C. P. Chen, H. Li, Y. Wei, T. Xia, and Y. Y. Tang, "A local contrast method for small infrared target detection," *IEEE transactions on geoscience and remote sensing*, vol. 52, no. 1, pp. 574–581, 2013.

[16] Y. Wei, X. You, and H. Li, "Multiscale patch-based contrast measure for small infrared target detection," *Pattern Recognition*, vol. 58, pp. 216–226, 2016.

[17] Y. Shi, Y. Wei, H. Yao, D. Pan, and G. Xiao, "High-boost-based multiscale local contrast measure for infrared small target detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 1, pp. 33–37, 2017.

[18] H. Deng, X. Sun, M. Liu, C. Ye, and X. Zhou, "Small infrared target detection based on weighted local difference measure," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 4204–4214, 2016.

[19] H. Deng, X. Sun, and X. Zhou, "A multiscale fuzzy metric for detecting small infrared targets against chaotic cloudy/sea-sky backgrounds," *IEEE transactions on cybernetics*, vol. 49, no. 5, pp. 1694–1707, 2018.

[20] P. Du and A. Hamdulla, "Infrared small target detection using homogeneity-weighted local contrast measure," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 3, pp. 514–518, 2019.

[21] Y. Dai, Y. Wu, F. Zhou, and K. Barnard, "Asymmetric contextual modulation for infrared small target detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 950–959.

[22] ——, "Attentional local contrast networks for infrared small target detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 11, pp. 9813–9824, 2021.

[23] L. Huang, S. Dai, T. Huang, X. Huang, and H. Wang, "Infrared small target segmentation with multiscale feature representation," *Infrared Physics & Technology*, vol. 116, p. 103755, 2021.

[24] B. Li, C. Xiao, L. Wang, Y. Wang, Z. Lin, M. Li, W. An, and Y. Guo, "Dense nested attention network for infrared small target detection," *IEEE Transactions on Image Processing*, vol. 32, pp. 1745–1758, 2022.

[25] T. Zhang, S. Cao, T. Pu, and Z. Peng, "Agpcnet: Attention-guided pyramid context networks for infrared small target detection," *arXiv preprint arXiv:2111.03580*, 2021.

[26] K. Wang, S. Du, C. Liu, and Z. Cao, "Interior attention-aware network for infrared small target detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–13, 2022.

[27] D. C. Stumpp, A. J. Byrne, and A. D. George, "Point-source target detection and localization in single-frame infrared imagery," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–11.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[30] Vck190 evaluation board user guide (ug1366). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1366-vck190-eval-bd

[31] Versal™ architecture and product data sheet: Overview (ds950). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/r/en-US/pg425-dpucv2dx8g

[32] M. Petry, P. Gest, A. Koch, M. Ghiglione, and M. Werner, "Accelerated deep-learning inference on fpgas in the space domain," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 2023, pp. 222–228.

[33] N. Perryman, C. Wilson, and A. George, "Evaluation of xilinx versal architecture for next-gen edge computing in space," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–11.

[34] S. R. Young, B. J. Steward, and K. C. Gross, "Development and validation of the afit scene and sensor emulator for testing (asset)," in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXVIII*, vol. 10178. SPIE, 2017, pp. 56–73.

[35] AFIT Sensor and Scene Emulation Tool (ASSET) [Computer Software]. Air Force Institute of Technology. [Online]. Available: www.afit.edu/CTISR/ASSET

[36] Vitis ai user guide (ug1414). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/

[37] Zcu104 board user guide (ug1267). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug1267-zcu104-eval-bd

[38] Zcu102 evaluation board user guide (ug1182). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug1182-zcu102-eval-bd

[39] Dpucv2dx8g for versal adaptive socs product guide (pg425). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/r/en-US/pg425-dpucv2dx8g

[40] Vck190/vmk180 board evaluation and management (beam) tool user guide (ug1573). AMD Xilinx. [Online]. Available: https://docs.xilinx.com/r/en-US/VCK190/VMK180-Board-Evaluation-and-Management-BEAM-Tool-User-Guide-UG1573