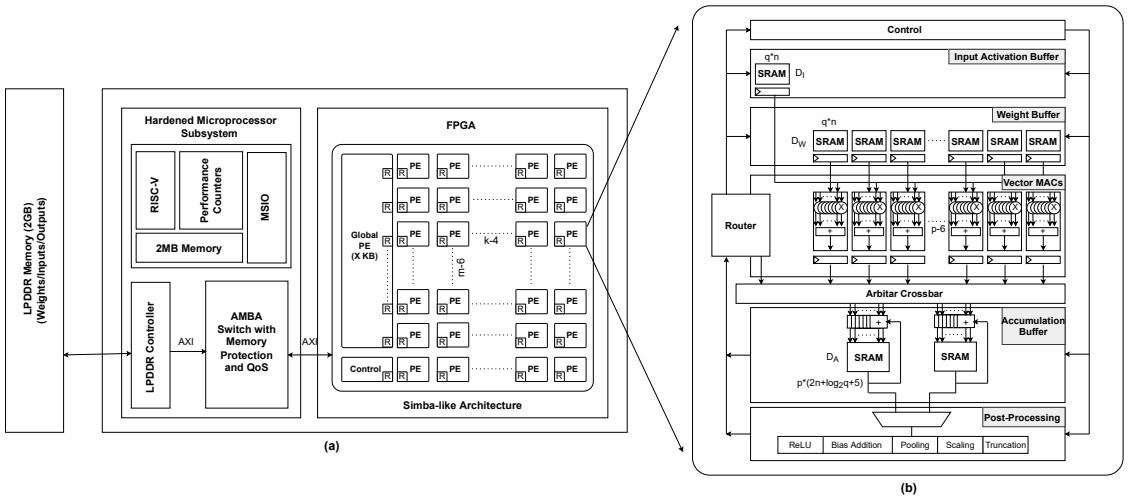
Fast and Efficient ML Hardware Accelerator Designs for SoCs with integrated FPGA

I. Overview of the idea

Machine Learning (ML) methods such as Deep Learning (DL) have shown promising results in real-time and edge applications for completing tasks such as image classification, object detection, speech translation, and language processing [1]. These DL algorithms are compute-intensive, and processing them using general-purpose compute cores do not usually meet the deployed application's operating constraints, such as latency and energy limits [2]. However, Convolutional Layers (CL) and Fully Connected Layers (FCL), which form the bulk of compute operations in DL algorithms, are highly parallelizable, have high data reuse and have deterministic operations [2], [3]. These properties of DL algorithms are exploited by ASIC accelerator designs such as Eyeriss [4] and Simba [3] for fast and efficient inference of DL workloads. However, specific ASIC implementations work best for only a particular set of workloads [2], and the design can not be *reshaped* on the fly to suit different types of DL workloads. This lack of flexibility is an issue in the following scenarios. First, the state-of-the-art (SOTA) algorithm for the specific tasks under consideration changes, and the existing organization of the accelerator is unfitting for the newer algorithm [1], [2]. Second, for systems that perform multiple tasks, such as drones that perform routing and obstacle avoidance during flight and reconnaissance of flood areas while hovering [5], a single DL accelerator might not be suitable for fast and efficient inferencing [6], [1]. Implementing suitable DL accelerator designs for the specific task or workload on an FPGA is a potential solution to overcome these challenges [6].



Parametrization in Simba-like Architecture Chiplet	
k	No. of PEs placed horizontally
m	No. of PEs placed vertically
X	Capacity of the Global PE (in KB)

Parameterization in PE	
n	Word Size (in bits)
p	Vector MAC count per PE
q	No. of multipliers per Vector MAC unit
D_I	Depth of Input Activation Buffer (in bits)
D_W	Depth of Weight Buffer (in bits)
D_A	Depth of Accumulation Buffer (in bits)

Fig. 1: Proposed implementation of Simba [3] like DL architecture on SoC integrated with FPGA for sensitivity analysis to identify design patterns best suited for FPGAs

The SOTA DL accelerator is Simba [3], and its internal organization is shown in Fig. 1. In the figure, the model (layer level input and weights) are fetched from the main memory and are stored at various levels of buffers such as global buffer, input buffer, and weight buffer of Processing Elements (PE) in the DL accelerator design. The input data is shared across multiple vector multiply and accumulate(MAC) units that operate on the weights of different output channels. Adder trees at each vector MAC accumulate the results into a single word, reducing writes to the accumulation buffer. Thus this SOTA architecture for ASIC achieves high parallelism while maintaining high efficiency due to reduced reads and writes across multiple PEs.

Nevertheless, implementing DL accelerators for fast and efficient inference on SoCs with FPGAs has its challenges. First, DL models stored in the system's main memory have to be brought into FPGA fabric over the shared memory controller. Hence, memory accesses by the accelerator must be minimized so as not to starve the host processor. Second, large buffers can be implemented on the FPGA fabric to reduce memory access to exploit data reuse in DL algorithms. However, this reduces the overall FPGA fabric that can be used for performing parallel compute operations. Third, MAC operations on DL accelerators are typically done at 8-bit precision [3], [2], and latency and energy values of MAC units implemented on FPGA are significantly affected by routing, in direct contrast to ASIC designs. Hence, all ASIC design patterns for DL accelerator design might not be suitable in the case of FPGAs.

In this work, we propose to identify the design patterns for fast and efficient inference DL accelerator implementations on FPGAs that address the above three significant challenges for various workloads. The proposed work is targeted towards vision

application, and the workloads under consideration are MobileNet, VGG, and ResNet. Toward this, we have identified the following architectural design parameters that would impact latency and energy. (i) Global Buffer size (ii) word size (iii) input, output, and weight buffer depths (iv) vector MAC width (v) vector MAC count in a PE (vi) Processing Element count (vii) spatial organization of PE. The impact of each of these parameters will be extensively studied and optimized for achieving maximum throughput and energy efficiency. We will also study the benefits on system latency and system energy when we implement high performance and approximate multipliers for FPGA-based hardware accelerators [7].

II. Implementation

Towards the above proposal, we implement the design shown in Fig. 1(b) on the PolarFire SoC and Nexys 4, and obtain latency, power, and resource utilization metrics. To identify the impact of various design choices on these metrics, realistic parameters for PE (n, p, q, D_I, D_W, D_A) are taken and varied for different instances. The results are given in the following section.

III. Result

A. Latency

The cycle time and the multiplier count directly affect the processing time of the neural network. In Fig. 2, the value of p×q gives the total number of parallel multipliers in the design. We observe that even if the total multiplier count is the same, the final latency can be different due to routing delay based on the arrangement inside the PE. We make the following observations; (i) For FPGA designs, the latency of a single vector MAC does not increase linearly with an increase in multiplier count until a threshold (16 in this case); Beyond the threshold, the increase in latency is sudden. (ii) Within the previously mentioned threshold, lower latency is achieved when aggregating multipliers at a single vector MAC rather than spreading it across multiple vector MACs.(iii) On increasing the vector MAC count by 8, the logic delay increases by only 2×. However, the routing delay increases by 11×. In other words, the routing delay starts dominating with an increase in vector MACs.

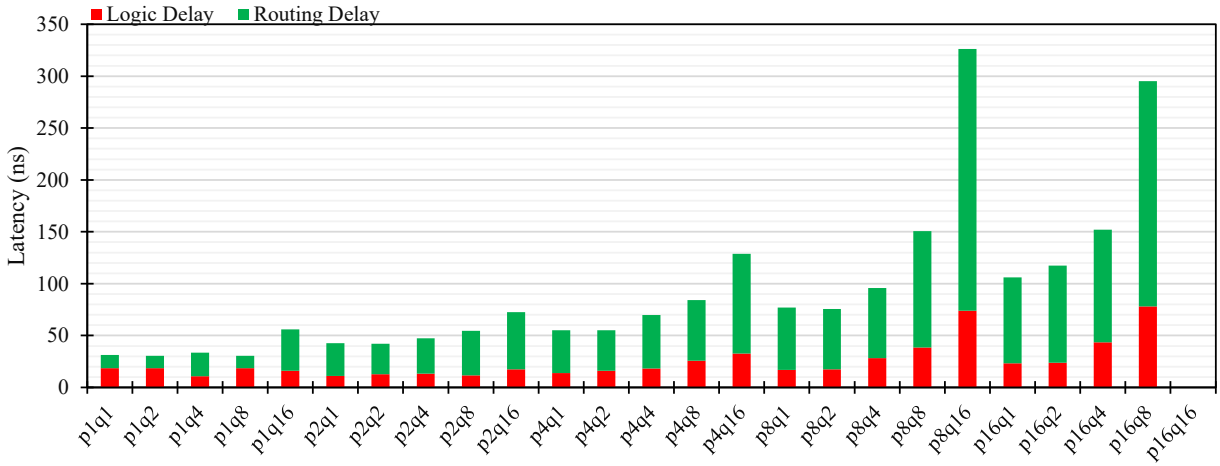


Fig. 2: Latency of the design for different PE configuration: Vector MAC Count (p) and number of multipliers per vector MAC (q)

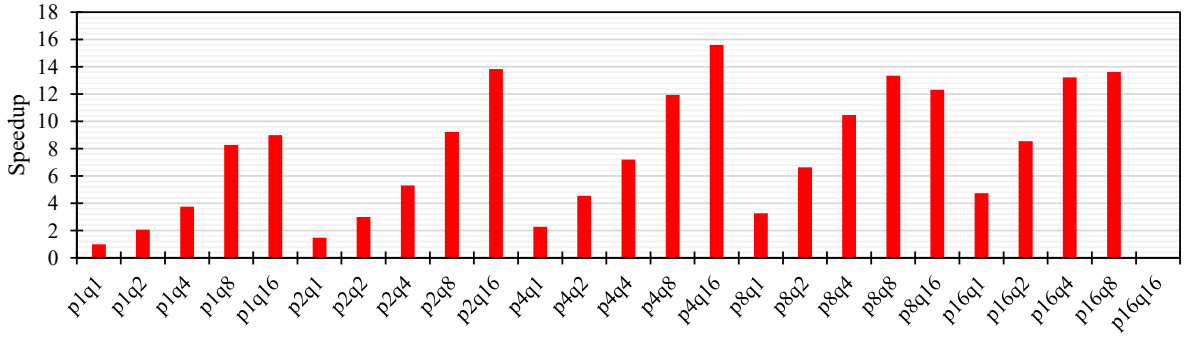


Fig. 3: Speedup of the design against p1q1 configuration

Further, from Fig. 3, which shows the effective speedup of each design, we can see that a design with a single vector MAC of size 8 is almost as fast as the 16 vector MACs with two multipliers each.

B. Power

The power consumed by the PE constrains the applications in which the accelerator can be deployed. Fig. 4 shows the power consumed by the PE under various configurations. We observe similar trends as that latency trends. We observe that (i) the increase in power is non-linear with an increase in multiplier count within a Vector MAC and remains fairly constant till it reaches a threshold. (ii) The Signal power starts dominating beyond a multiplier count of 16.

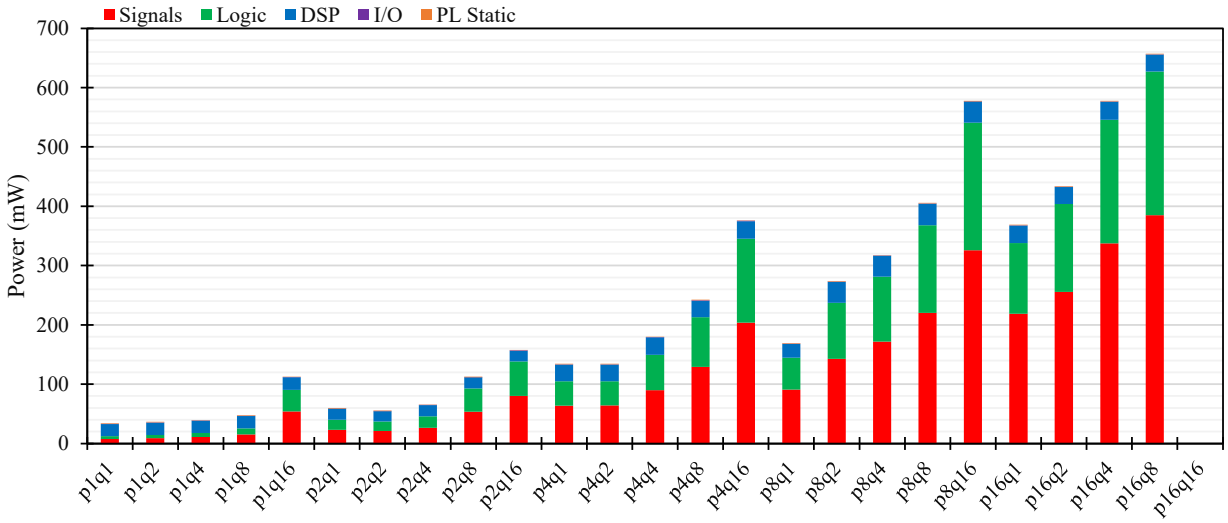


Fig. 4: Power of the design for different PE configuration: Vector MAC Count (p) and number of multipliers per vector MAC (q)

C. Resource Utilization

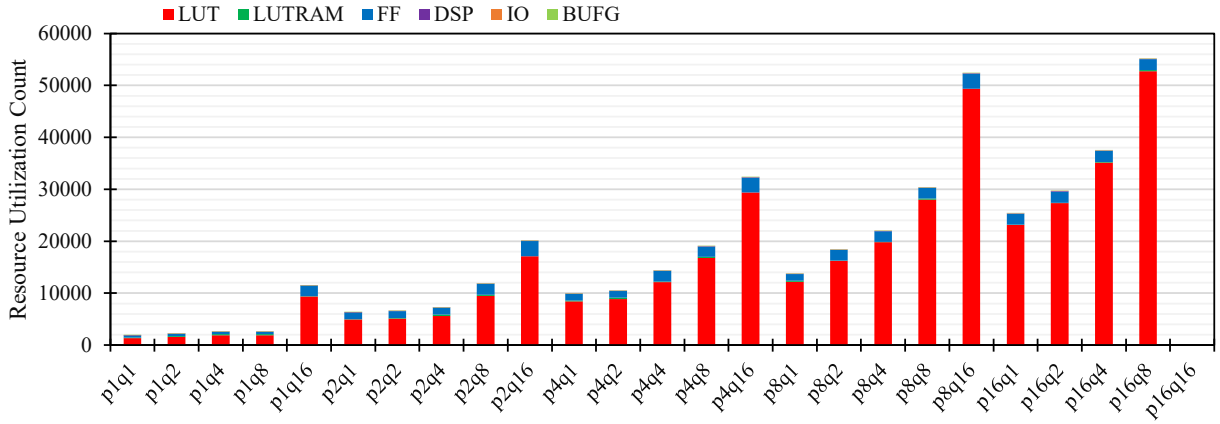


Fig. 5: Resource Utilization of the design for different PE configuration: Vector MAC Count (p) and number of multipliers per vector MAC (q)

Fig. 5 shows the count of various resources used for implementing each PE design. We observe that when doubling the multiplier count per vector MAC, the LUT utilization per multiplier decreases linearly. Further, at a higher multiplier count (For example, p4p4 vs. p8q4), a design with twice the number of multipliers uses the same LUT count.

IV. Conclusion

When designing DNN accelerators using FPGA with vector MACs inside the Processing Element, having wider vector MACs within a threshold will give lower power and latency. Whereas using narrower vector MACs and increasing the vector MAC count decreases the LUT utilization.

V. Team Members and Contact Details

Ruchit Chudasama, Daniel Giftson, Tom Glint, Aryan Gupta, Joycee Mekie, Sukanya More, Vrajesh Patel, Ashiwini Pathak, Kailash Prasad, Prateek Sharma, Megha Yadav
Indian Institute of Technology Gandhinagar
India
{ruchit.chudasama, daniel.giftson, tom.issac, aryan.gupta, joycee, sukanya.more, patel.vrajesh, pathakashiwini, kailash.prasad, sharmaprateek, yadavmegha}@iitgn.ac.in

Corresponding Author - Vrajesh Patel (patel.vrajesh@iitgn.ac.in)
Name of University - Indian Institute of Technology Gandhinagar.
Shipping Address - K-437, Kyzeel Hostel, IIT Gandhinagar, Palaj Village, Gandhinagar, Gujarat - 382355, India.
Contact Number - 8154054997.

References

[1] V. J. Reddi, C. Cheng *et al.*, "Mlperf inference benchmark," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 446–459.
[2] Y. Chen, Y. Xie *et al.*, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.
[3] B. Zimmer, R. Venkatesan *et al.*, "A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.
[4] Y.-H. Chen, T. Krishna *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
[5] V. Chamola, V. Hassija *et al.*, "Disaster and pandemic management using machine learning: a survey," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 16 047–16 071, 2020.
[6] C. Wu, V. Fresse *et al.*, "Accelerating dnns from local to virtualized fpga in the cloud: A survey of trends," *Journal of Systems Architecture*, vol. 119, p. 102257, 2021.
[7] S. Ullah, S. Rehman *et al.*, "High-performance accurate and approximate multipliers for fpga-based hardware accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211–224, 2022.