NetPU: Prototyping a Generic Reconfigurable Neural Network Accelerator Architecture

Yuhao Liu, Shubham Rai, Salim Ullah, Akash Kumar
Chair of Processor Design, Center for Advancing Electronics Dresden (CfAED), TU Dresden, Germany
Email: {yuhao.liu1, shubham.rai, salim.ullah, akash.kumar}@tu-dresden.de

2022 International Conference on Field-Programmable Technology (ICFPT) | 978-1-6654-5336-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICFPT56656.2022.9974206

I. Introduction

FPGA-based Neural Network (NN) accelerator is a rapidly advancing subject in recent research. Related works can be classified as two hardware architectures: i) ***Heterogeneous Streaming Dataflow*** **(HSD)** architecture and ii) ***Processing Element Matrix*** **(PEM)** architecture. HSD architecture explores the reconfigurability of FPGAs to support the customization and optimization of hardware design to implement a complete network on FPGA for one given trained model. PEM architecture achieves relatively generic support for different network models, essentially implementing the neuron processing modules on the FPGA scheduled by the runtime software environment. In summary, the HSD architecture requires more resources with simplified runtime software control. The PEM architecture consumes fewer resources than the HSD architecture. However, the runtime software environment can be a heavy payload for lightweight systems, such as the low-power microcontroller of IoT or edge devices.

Our work explores a new hybrid architecture, ***NetPU***, based on the above two architectures. This architecture is designed to achieve the generic support for different NN models as PEM architecture with a simplified runtime software environment as HSD architecture. Our ***NetPU*** implementations are created by Verilog on Vivado. As shown in Figure 1, this work consists of a three-stage structure: ***Network Processing Unit*** **(NetPU)**, ***Layer Processing Unit*** **(LPU)**, and ***Neuron Processing Unit*** **(NPU)**. NetPU reuses LPUs and NPUs to extend the flexibility of accelerators fitting different sizes of network model workloads. This accelerator saves the required parameters of current accelerating layers in LPUs to on-chip memories, reducing the storage pressure for loading complete network parameters of large models. This architecture supports scalable, mixable (the data precision of different layers can be different), quantized (1-8 bits) precision, BN folding, and selectable activation functions, including *ReLU*, *Piecewise Linearized Sigmoid/Tanh* [1], *Sign*, and *Multi-Thresholds* [2]. All the above accelerator configurations can be reconfigured in runtime without changing hardware for different models. Figure 2 shows the hardware design of our runtime-reconfigurable NPUs: crossbar schedules the data

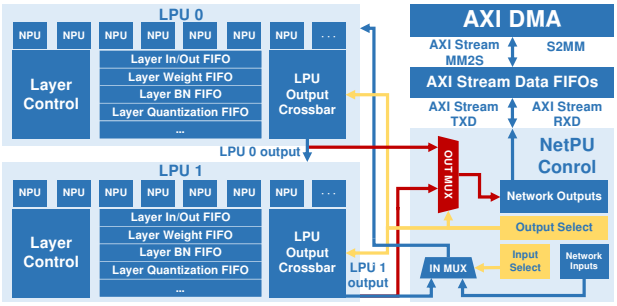

Fig. 1: Hardware Structure of NetPU

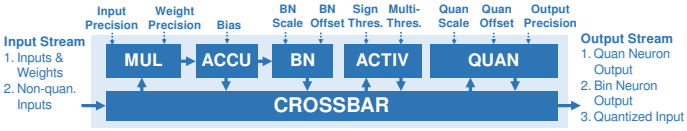

Fig. 2: Hardware Design of the runtime reconfigurable NPUs on *Completely Generic Mode*

stream to bypass some submodules to support different network models. For further reducing the on-chip hardware resource consumption, we also offer two other simplified versions of NetPU, *Quantization Generic Mode* (QGM) and *Binarization Generic Mode* (BGM) for special application scenarios. The above two modes fix the precision, activation function, and BN folding setting in NetPU architecture but still do not limit the scale of network models.

II. Simulation and Evaluation

Table I lists the simulation results of our work and the comparison with FINN instances [3]. We applied the pre-trained network models from the FINN project with three sizes — TFC (64×3), SFC (128×3), and LFC(1024×3) in our simulation. Results show that, in *Completely Generic Mode* (CGM), our work can support different network models with different sizes and precision without hardware regeneration. Regarding the QGM and BGM instances, resource consumption can be further reduced. As compared to the FINN instance, our work consumes fewer LUTs and BRAMs resources. However, our current work has high latency as a trade-off of generic support in network inference. Besides, we actually measured a 2-bit quantized tiny size model (TFC, 64x3, W2A2) in a CGM-mode instance on Ultra96 V2 hardware. The average latency of this instance is 181.74us. The power report generated after the implementation of CGM mode shows the total on-chip power is 2.313W.

TABLE I: Simulation of NetPU on Ultra96-V2 Compared with FINN Instances

Work	Implementation	Target Platform	Clock (MHz)	Resource Utilization LUT	BRAM	DSP	Latency (μs) 64×3	256×3	1024×3
NetPU	BGM-32	Ultr96 V2	100	20,932	45.5	-	60.645	248.805	1923.045
	BGM-64			56,776	45.5	-	41.765	139.685	992.165
	QGM-32			19,035	49.5	352	104.225	485.105	3851.825
	CGM-64			59,755	129.5	352	38.745 (***W1A1***)	133.785 (***W2A2*** and ***W2A1***)	974.745
							172.165	882.085	7408.225
FINN	SFC-max	Zynq7000	200	91,131	4.5	-	-	0.31	-
	LFC-max			82,988	396	-	-	-	2.44
	SFC-fix			5,155	16	-	-	240	-
	LFC-fix			5,636	114.5	-	-	-	282

References

[1] Hesham Amin, K Memy Curtis, and Barrie R Hayes-Gill. "Piecewise linear approximation applied to nonlinear function of a neural network". In: *IEE Proceedings-Circuits, Devices and Systems* (1997).
[2] Zhaowei Cai et al. "Deep Learning with Low Precision by Half-wave Gaussian Quantization". In: *CoRR* (2017).
[3] Yaman Umuroglu et al. "Finn: A framework for fast, scalable binarized neural network inference". In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2017.