# Modelling and Analysis of FPGA-based MPSoC System with Multiple DNN Accelerators

Cong Gao, Xuqi Zhu, Sangeet Saha, Klaus D McDonald-Maier and Xiaojun Zhai
*University of Essex*, Colchester, United Kingdom
{cg21670, xz18173, sangeet.saha, kdm, xzhai}@essex.ac.uk

***Abstract*—Deep Neural Networks (DNNs) have been widely applied in many fields for decades, and a standard method for deploying them on embedded systems involves using accelerators. However, due to the resource constraints of embedded systems, improving energy and computing efficiency becomes one of the research challenges in this domain. DNN model optimization and NAS (Neural Architecture Searching) are commonly used to strengthen the DNN model running efficiency on an embedded system. However, because the system's runtime workloads are varied in practical situations, to further improve the computing efficiency of the system at runtime, real-time hardware and software design space exploration is required to ensure the system is running at the optimal time state at runtime. This paper presents a comprehensive modelling and analysis approach for the performance data (e.g., latency, energy consumption, accuracy, etc.) collected from an AMD-Xilinx heterogeneous MPSoC platform equipped with multiple DNN accelerators. The results demonstrate that the relationships between accuracy loss, hardware performance, and model size are significantly correlated. Furthermore, an appropriate hardware and software configuration could be obtained by giving constraints at runtime.**

***Index Terms*—FPGA, Heterogeneous embedded systems, MPSoC, Deep Neural networks, Edge computing, Energy efficiency**

## I. Introduction

With the growth of depth and complexity of the interconnections, the Deep Neural Networks (DNN) obtains increasing inference accuracy on complex tasks, such as image classification and speech processing [1]. The GPT [2] demonstrates a considerable model with billions of parameters that can answer sophisticated questions like humankind. However, achieving excellent inference performance relies on significant computational resources and vast amounts of data [3].

DNN-based AI applications show attractive potential across a wide range of fields. Especially for edge computing, the millions of edge equipment generate enormous data every second [3]. The conventional solution is to adopt cloud services to support a part or all of the inference mission. But given the massive growing throughput from the edge and the demands of sensitive data security, deploying AI applications on edge devices is an alternative solution [3]. However, the considerable run-time computational resources cost is an inevitable challenge for employing DNN on resource-limited edge equipment. Also, with the fast-growing amount of edge computing equipment, energy costs related to carbon dioxide emission are an essential issue to consider.

To implement DNNs on resource-constrained embedded systems with lower energy costs, pruning and quantizing are mostly adopted optimization methods to discard redundant operations from the model and release computational resources [3], [4]. Modifying the network structure and the data structure with an acceptable accuracy loss on the embedded system can reduce the models' size, and the hardware budget will reduce simultaneously. Furthermore, the inference tasks can be executed on resource-limited edge devices with tolerable accuracy loss by deploying the optimized network.

The other approach, Neural Architecture Searching (NAS), finds neural networks compatible with given hardware instead of optimizing the existing network [5]. Recently, the OFA (once for all) framework proposed by [6] has become one of the most successful NAS solutions, allowing developers to find an optimal network among $2 \times 10^6$ sub-networks for a specific hardware platform.

Both NAS and network optimization concentrate on software implementation and neglect to optimize the hardware since most conventional edge devices use fixed hardware architecture. To optimize hardware, the Application Specific Integrated Circuit (ASIC) and Field-Programmable Gate Array (FPGA) are introduced [4]. Compared with ASIC, FPGA is more flexible because of its programmable architecture ability. This ability makes FPGA an affordable solution for rapid technology evolution. In addition, the Dynamic Partial Reconfiguration (DPR) technology [7] makes it possible to adopt run-time optimization strategies to reallocate computing resources and memory access paths in the FPGA platform. [8] proposed a convolution processing unit by employing DPR to achieve adaptive precision in the run-time. In contrast, an application-level resource scheduling strategy provided by [9], [10] uses DPR to accelerate DNNs and achieve higher performance against a constrained resource budget.

In the design stage, considering hardware and neural networks, a so-called co-design design flow is widely used in an edge computing system. Researchers propose methodologies to design an optimal hardware design with a particular neural network [11]–[13]. However, in a practical edge computing scenario, the system's workload can be varied due to the uncertainty of the input environment parameters. Taking a traffic

This work is supported by the UK Engineering and Physical Sciences Research Council through grants, EP/V034111/1, EP/X015955/1, EP/X019160/1 and EP/V000462/1. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

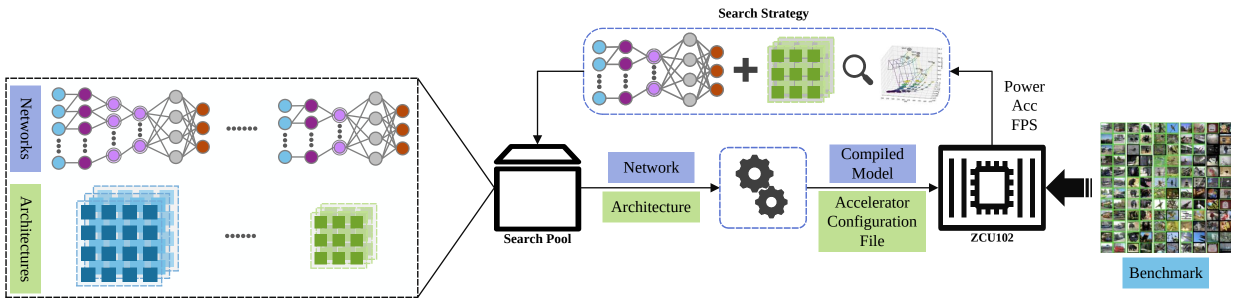


Fig. 1. Overview of the experiment setup

control system as an example, the input varies significantly in terms of the number of vehicles and pedestrians, a fixed optimal solution would either be too energy-intensive or fail to meet the desired time latency. Therefore, the optimal solution needs to adjust its computing resources according to the input loads to improve energy efficiency.

In such real-time scenarios, a single optimal solution is often insufficient, and dynamic solutions are thus needed. A large amount of design work is required beforehand. An adaptive scheme to schedule the hardware configuration and corresponding neural network model meeting an essential latency requirement and achieving better energy efficiency is critical.

This paper aims to find the hardware and neural network model pair from various configurations based on required time constraints. This optimal pair costs less energy while meeting the time constraint in a specific application/scenario. To achieve that, a modelling and analysis approach to show the relationships between accuracy loss, hardware performance, and scale of a DNN model is set up based on the data collected from experiments on the AMD-Xilinx ZCU102 MPSoC. These models can be later used to determine an optimal choice for achieving better computing and energy efficiency at runtime.

## II. Experiment Overview

This section mainly explains the experiment setup for obtaining various hardware and software metrics (e.g., power consumption, processing speed, model accuracy, etc.) of multiple DNN accelerators running on an FPGA-based MPSoC System. The ultimate goal is to help us to analysis an optimal pair under various constraints. Fig. 1 represents the overview of the experimental setup for the data collection flow. First, a sub-model is selected and deployed using one of the hardware configuration settings from the predesigned model and hardware configuration pool, considered a model-hardware pair. Each pair is then tested on hardware using the same image classification benchmark. During the test, we record the processing speed, model accuracy and power consumption, and every 60s, the model and hardware are reconfigured at runtime. The hardware reconfiguration uses the DFX technology (dynamic function exchange), where only the hardware accelerators in the Reconfigurable Partition (RP) are configured. The rest of the systems are running as normal, with a minimal distribution of the processing pipeline.

### A. Experimental platform and DNN hardware accelerator

This experiment uses an AMD-Xilinx ZCU102 platform equipped with Zynq UltraScale+ MPSoC (XCZU9EG). For the DNN hardware accelerator, we use the Deep Learning Processing Unit (DPU), a soft IP core to enable deep learning consumer design and set up AI projects on AMD-Xilinx FPGA. The DPU IPs are formed with different settings, detailed in the data sheet [14]. In this paper, we configure 7 sets of DPUs, each consisting of two same DPU configurations. The resource utilization for each set of DPU configurations is presented in Table. I. The hardware design is implemented using Vitis (2022.1) and Xilinx Runtime Library (XRT) (2022.1).

### B. DNN models

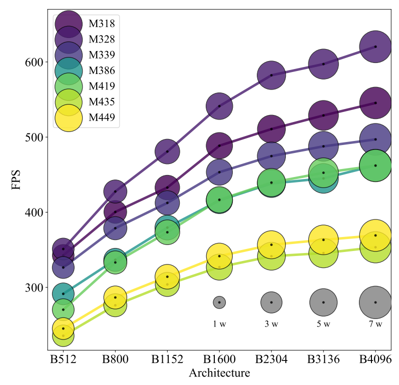
OFA is a neural architecture search (NAS) technique for efficient model design proposed by [15], which provides a possible way to produce many submodels from a large "super-network" that contains multiple sub-networks with different widths, depths, and resolutions. The OFA uses pruning and scaling techniques to generate a family of smaller, more efficient sub-networks tailored to specific tasks. With an adjustable parameter on depth, width, and kernel size, it can produce nearly $2 \times 10^6$ sub-networks based on one super-network. In this paper, we use the OFA to produce 7 versions of Resnet50 with different FLOPs (Floating Point Operations). Combining them with 7 different DPU architectures will test $7 \times 7 = 49$ different configurations. To compile the DNN models on the target hardware platform, we use Vitis AI (2.5) [16].

## III. Modelling and Analysis

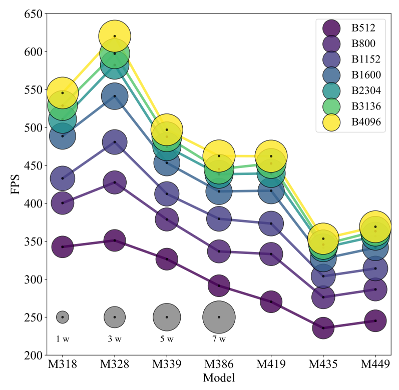
In this section, a comprehensive comparison and modelling analysis is performed for the obtained experiment data, where the relationship between model, hardware, and processing, energy consumption are carefully explored by analyzing the data from different points of view.

### A. Model accuracy and Flops

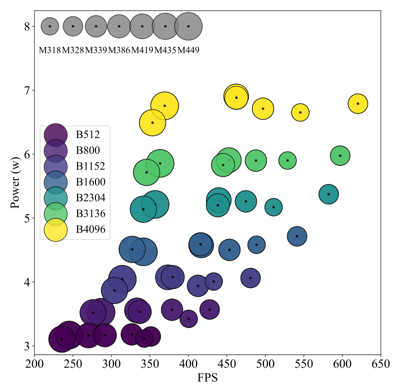
In our experiment, around 50 submodels are generated from the super-network via the OFA algorithm, as shown in Fig.2(f). Normally, a bigger model can achieve better accuracy



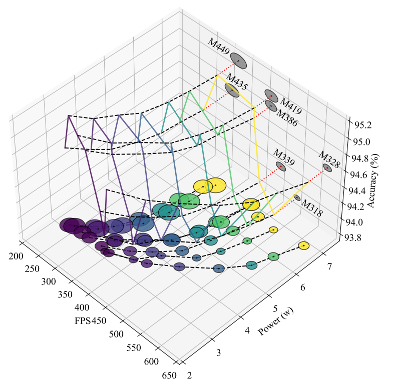
(a)



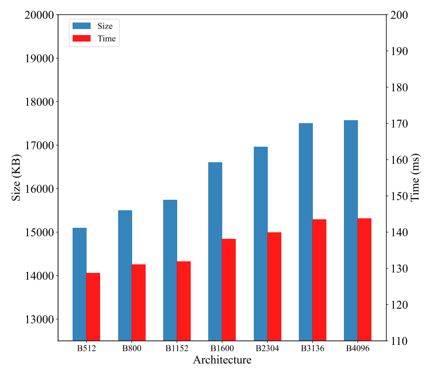
(b)



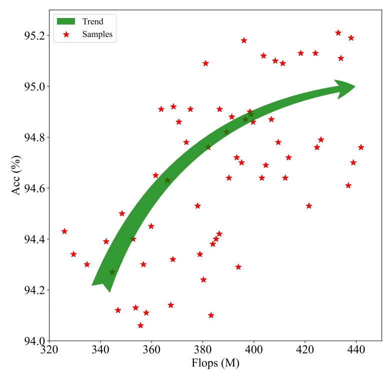
(c)



(d)



(e)



(f)

Fig. 2. The scatter plots for different dimensions, models, hardware, and performance. (a) Power and FPS for different hardware architectures (Model view); (b) Power and FPS for different DNN Models (Architecture view); (c) FPS and power consumption for different models and architectures (Performance view); (d) FPS, power consumption and accuracy of different models and architectures (3D Performance view); (e) DFX configuration times for different architectures; (f) FLOPs and accuracy comparison of DNN models

TABLE I
ZCU102 UTILIZATION WITH DIFFERENT SETTINGS OF DPUS

| Hardware Setting | LUT (%) | Register (%) | Block RAM (%) | DSP (%) |
|---|---|---|---|---|
| $B512 \times 2$ | 19.64 | 12.6 | 15.79 | 9.37 |
| $B800 \times 2$ | 21.69 | 15.01 | 19.74 | 13.17 |
| $B1024 \times 2$ | 24.86 | 17.53 | 22.81 | 18.25 |
| $B1152 \times 2$ | 23.74 | 17.28 | 26.54 | 17.62 |
| $B1600 \times 2$ | 28.03 | 21.46 | 27.63 | 25.87 |
| $B2304 \times 2$ | 30.74 | 25.11 | 36.18 | 34.76 |
| $B3136 \times 2$ | 34.09 | 29.08 | 45.61 | 44.92 |
| $B4096 \times 2$ | 38.06 | 35.85 | 55.92 | 56.35 |

in Fig.2(f). However, many points (models) also have better accuracy but with a smaller size when compared with points nearby. This is mainly because the model size was not set as the primary network searching criteria.

## B. Cost of run-time reconfiguration

With the technology of DFX, hardware settings can be reconfigured in real-time to achieve an adaptive DNN-based hardware system according to the environment. However, it takes ms level time latency to achieve the reconfiguration work, and the cost is varied based on the type of different hardware settings. The time cost mainly depends on the size/scale of the reconfigured region and bitstream. By using the XRT(Xilinx Runtime) API, the reconfigured program can be quickly compiled and run on the processing system. The time cost for each hardware setting switch is measured, recorded, and presented in Fig. 2(e).

## C. Fixed model & hardware configurations

In Fig. 2(a), each line represents the FPS results obtained from different DPU architectures using the same DNN model (e.g. M318, M328, etc.). For example, M318 means a DNN model contains 318M FLOPs. In this experiment, we applied different DNN models on different DPU architectures and recorded the FPS and power consumption as shown in Fig. 2(a). In this figure, each circle represents the power consumption and FPS results during the benchmark testing. The diameter of each circle distinguishes values of the power consumption, the larger diameter is, the more energy is consumed. In general, using the same neural network model, allocating more hardware resources will increase the data processing speed at the cost of more power consumption.

In Fig. 2(b), while analyzing the issues in another dimension, hardware, each line represents a set of hardware configurations. With the same hardware setting, a larger DNN model will take more time to process, but it may achieve better accuracy with a slight cost on power consumption.

### D. Discussion

Hardware configurations and model sizes are two major factors that have positive impacts on the FPS but they will cost more energy. Usually, a model with large FLOPs contains more parameters and a more complex architecture, thus normally resulting in a larger model size; Also, because the model utilises more calculation resources, it takes more time to process the data resulting in a lower FPS. However, there exist some special cases, for example, the size of model M328 is smaller than the M318, which results in a higher FPS.

### E. Optimal hardware-model selection

To achieve the necessary processing speed and accuracy while minimizing power consumption, it is important to select the most suitable hardware-model combination. This selection should be optimal for the desired outcome. For example, a practical real-time DNN-based video processing application on edge devices has a basic requirement for the processing speed of real-time video processing tasks, and it also wants to achieve better energy efficiency.

In Fig. 2(c), a single circle represents one set of hardware and DNN models combination. We can consider the dataset as mapping relationships below:

$(Model_x, Hardware_y) \longleftrightarrow (Power, FPS)$
$(Model_x) \longleftrightarrow (Accuracy)$

Each set of model-hardware combination links to a set of power and FPS metrics, and each model has an accuracy value:

$$Power_{x,y}, FPS_{x,y} = f(Model_x, Hardware_y) \tag{1}$$
$$Accuracy_x = g(Model_x) \tag{2}$$

One of our objectives is to select the appropriate set of models and hardware combinations to minimise power consumption and achieve minimal FPS requirements with a relatively less cost on accuracy loss when switching to a small size of DNN model. In order to mathematically formulate the problem, we define the following variable $Z_{x,y}$ which becomes 1 if the $(x, y)$ is selected. so, with the variable $Z_{x,y}$, the optimal searching strategy can be described in the following equations set:

$$Minimize \quad Power_{x,y} \times Z_{x,y} \quad \forall x, y \tag{3}$$

subject to:

$$FPS_{x,y} \times Z_{x,y} \geq FPS_{target} \quad \forall x, y \tag{4}$$
$$DF_{x,y} \times Z_{x,y} \leq DF_{target} \quad \forall x, y \tag{5}$$

where:

$$Z_{x,y} = 1 \quad if \quad (x, y) \quad is \quad selected \tag{6}$$
$$DF_{x,y} = \frac{Accuracy_x - Accuracy_{x-1}}{Power_{x,y} - Power_{x-1,y-1}} \tag{7}$$
$$DF_{target} = \frac{Accuracy_{max} - Accuracy_{min}}{Power_{max} - Power_{min}} \tag{8}$$

## IV. Conclusion

While deploying DNNs on edge devices, reducing energy costs is an essential issue that needs to be solved. As for the uncertainty of the real-time environment parameters, the system's workload varies; Then, an adaptive edge computing embedded platform will select a suitable hardware and DNN model combination to fulfill the targeted requirement. In this paper, modelling and analysis are established based on the experiment setup, which can be used to find an optimal hardware and model combination to achieve a relatively lower energy cost when satisfying the time latency and accuracy of the system at runtime. The hardware and model selection methods are still limited, so optimal decisions can be easily selected according to a predefined optimisation scheme. Our future work will vastly increase the diversity of hardware and model configurations, and develop a multi-parameter searching algorithm, in order to predict optimal hardware and model settings to achieve better computing and energy efficiency for accelerating DNN on edge devices.

## References

[1] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *ACM SIGPLAN Notices*, vol. 53, no. 6, pp. 31–43, 2018.

[2] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman, "WebGPT: Browser-assisted question-answering with human feedback," 2021. [Online]. Available: http://arxiv.org/abs/2112.09332

[3] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[4] K. S. Zaman, M. B. I. Reaz, S. H. M. Ali, A. A. A. Bakar, and M. E. H. Chowdhury, "Custom Hardware Architectures for Deep Learning on Portable Devices: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[5] L. Tang, H. Li, C. Yan, X. Zheng, and R. Ji, "Survey on neural architecture search," *Journal of Image and Graphics*, vol. 26, no. 2, pp. 245–264, 2021.

[6] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize it for Efficient Deployment," pp. 1–15, 2019. [Online]. Available: http://arxiv.org/abs/1908.09791

[7] M. Nguyen, N. Serafin, and J. C. Hoe, "Partial Reconfiguration for Design Optimization," *Proceedings - 30th International Conference on Field-Programmable Logic and Applications, FPL 2020*, no. 1, pp. 328–334, 2020.

[8] D. Cain, O. Eldash, K. Khalil, and M. Bayoumi, "Convolution Processing Unit Featuring Adaptive Precision using Dynamic Reconfiguration," *7th IEEE World Forum on Internet of Things, WF-IoT 2021*, pp. 592–597, 2021.

[9] Y. Lu, C. Gao, R. Saha, S. Saha, K. D. McDonald-Maier, and X. Zhai, "Fpga-based dynamic deep learning acceleration for real-time video analytics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13642 LNCS, pp. 68–82, 2022.

[10] C. Gao, S. Saha, Y. Lu, R. Saha, K. D. Mcdonald-Maier, and X. Zhai, “Deep learning on fpgas with multiple service levels for edge computing,” *2022 27th International Conference on Automation and Computing: Smart Systems and Manufacturing, ICAC 2022*, 2022.

[11] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-m. Hwu, and D. Chen, “Fpga/dnn co-design: An efficient design methodology for 1ot intelligence on the edge,” in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[12] X. Zhang, Y. Ma, J. Xiong, W.-M. W. Hwu, V. Kindratenko, and D. Chen, “Exploring hw/sw co-design for video analysis on cpu-fpga heterogeneous systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1606–1619, 2022.

[13] J. Haris, P. Gibson, J. Cano, N. B. Agostini, and D. Kaeli, “Secda: Efficient hardware/software co-design of fpga-based dnn accelerators for edge inference,” in *2021 IEEE 33rd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2021, pp. 33–43.

[14] Xilinx, “Dpuczdx8g for zynq ultrascale+ mpsocs product guide (pg338),” Tech. Rep., 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/pg338-dpu

[15] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, “Once for all: Train one network and specialize it for efficient deployment,” in *International Conference on Learning Representations*, 2020. [Online]. Available: https://arxiv.org/pdf/1908.09791.pdf

[16] Xilinx, “Vitis AI User Guide (UG1414),” Tech. Rep., 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview