Design Space Exploration of FPGA Based System with Multiple DNN Accelerators

Rajesh Kedia, Shikha Goel, M. Balakrishnan, Kolin Paul, and Rijurekha Sen

***Abstract*—Many emerging systems concurrently execute multiple applications that use deep neural network (DNN) as a key portion of the computation. To speed up execution of such DNNs, various hardware accelerators have been proposed in recent works. Deep Learning Processor Unit (DPU) from Xilinx is one such accelerator targeted for FPGA based systems. We study the runtime and energy consumption for different DNNs on a range of DPU configurations and derive useful insights. Using these insights, we formulate a design space exploration (DSE) strategy to explore trade-offs in accuracy, runtime, cost, and energy consumption arising due to flexibility in choosing DNN topology, DPU configuration, and FPGA model. The proposed strategy provides a reduction of 28× in the number of design points to be simulated and 23× in the pruning time.**

***Index Terms*—Design space exploration, Deep neural networks, FPGA, Accelerators, Embedded systems.**

I. Introduction

Many energy and runtime efficient deep neural network (DNN) accelerators have been proposed recently [1]–[4]. Xilinx's DPU (Deep Learning Processor Unit) [3], [4] is a generic DNN accelerator which can be used for any DNN topology (unlike others that are specific to particular DNN [1], [2]). DPU can be configured for different sizes that vary in terms of resource requirements and execution rate. Further, many systems [2], [5], [6] have a growing need to support multiple concurrent applications that involve DNN computations.

Each of these applications can be mapped to one or more standard DNNs (different topology) with different inference accuracy [7]. Due to the generic nature of DPU, each choice of DNN for an application can be mapped to a different DPU size, resulting in different execution times. Further, the count and size of DPUs that can be integrated on an FPGA depends upon the resources available on the chosen FPGA chip which in-turn influences the DNNs that are feasible within a required execution time period. Such dependencies of applications on DNN topology and DPU size and then further on the chosen FPGA chip creates a complex design space. In this letter, we explore this design space at an early stage of design cycle to define the hardware platform that can support the required accuracy level, cost, and energy budget.

We study the performance and energy behavior of different DNNs on various DPU sizes using FPGA board and derive insights resulting in pruning rules for DSE (Section IV). We obtain 28× reduction in the number of invocations of the simulator/estimation function and an overall speedup of 23× in exploration time. Our DSE flow also identifies newer Pareto points (Fig. 5) in comparison to just using a fixed type of FPGA/platform. This letter makes the following contributions:

1) The first work to consider DSE at an early stage for DNN accelerator (DPU) based inference system.
2) DSE strategy involving various designer insights derived using measurements from a real FPGA board.
3) Comprehensive results demonstrating benefits of the proposed approach on standard DNNs.

II. Related Work

A system with multiple types of DPU can be considered as a heterogeneous multi-core system. Baruah [8] was one of the firsts to study the feasibility of given real-time tasks on heterogeneous systems. Recently, Chwa et al. [9] and Moulik et al. [10] extended Baruah's work [8] to consider task migration and minimize preemption. All these works focus on mapping of tasks onto given cores on the platform. However, they neither explore allocating the number/type of cores nor consider different accuracy levels for the tasks.

Quan et al. [11] and Stralen et al. [12] identify appropriate platform to be allocated and task mapping for a scenario aware system. Führ et al. [13] address hardware-software partitioning of tasks onto a Xilinx Zynq device. Li et al. [14] explore pre-characterized DSP function library for FPGA to explore area and latency trade-offs. However, none of these works consider tasks with variable accuracy levels. Quan et al. [11] and Stralen et al. [12] consider resource allocation but are limited to identifying whether to include a particular resource or not. Exploring the number/type of each resource is a more complex problem, which is not addressed in these works.

Network Architecture Search (NAS) [15] explores the internal architecture of DNN for a single application. For an advanced driver-assistance system, Peng et al. [5] proposed combined training to use one DNN for all inference tasks. Both these approaches are orthogonal to our focus of efficiently using off-the-shelf DNNs and components. Studying co-execution of multiple DNN accelerators has experienced limited attention. f-CNN^x [2] (a recent work) maps multiple independent CNN applications on a given FPGA; however, it neither explores choice of FPGA nor trade-off in accuracy.

III. System Model and Problem Statement

A. Notations

$\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$ is a set of n periodic applications repeating at a defined interval known as period. Each application $A_i = \langle\langle a_i^1, a_i^2, \ldots, a_i^s\rangle, p_i\rangle$, where p_i represents the period

The authors are affiliated to the Indian Institute of Technology Delhi, India. E-mail: {kedia, shikha.goel, mbala, kolin, riju}@cse.iitd.ac.in.

Lokesh Siddhu provided useful feedback on the manuscript. Rajesh Kedia was supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India MEITY-PHD-2671. MeitY also funded ZCU102 board under "SMDP-C2SD" project.

of A_i and a_i^j represents the accuracy of application A_i when mapped to network (DNN) N^j. If application A_i cannot use N^j, then $a_i^j = 0$.

$\mathbf{N} = \{N^1, N^2, \dots N^s\}$ is a set of s different networks. Each network N^j is characterized as: $N^j = \langle t_1^j, t_2^j, \dots, t_m^j \rangle$ where t_k^j represents the execution time when network N^j is executed on DPU of type D_k. Similar model has been used previously for heterogeneous multi-core systems [8], [10].

$\mathbf{D}$ is an ordered set ($\{D_1, D_2, \dots, D_m\}$) of m different types of DPUs available for implementation on FPGA. Each DPU type is characterized as $D_k = \langle area_k, perf_k \rangle$, where $area_k$ is a tuple representing the area (number of FPGA resources in terms of LUT, BRAM, and DSP) required for D_k and $perf_k$ is the peak performance in GOPs (Giga operations per second) supported by D_k. $\mathbf{D}$ is ordered in increasing order of $perf$ (and $area$ as well). The final hardware platform consists of a few of such DPU IPs realized on an FPGA.

$\mathbf{F} = \{F_1, F_2, \dots, F_r\}$ is a set of r FPGA chips. Each FPGA chip $F_l = \langle areaTotal_l, cost_l, f_l \rangle$, where $areaTotal_l$ represents the total resources (in terms of LUT, BRAM, and DSP), $cost_l$ represents the price, and f_l represents the normalized frequency of the FPGA w.r.t. a reference FPGA (ZCU102). f_l is used for scaling the execution time for different FPGAs (currently only Zynq devices support Xilinx DPU).

B. Problem Statement

The design space being explored in this letter consists of various application ($\mathbf{A}$) to DNN ($\mathbf{N}$) and DNN ($\mathbf{N}$) to DPU ($\mathbf{D}$) mappings. It considers count of each DPU type (D_k) to be used, mapping of applications on allocated DPUs, and the FPGA chip (F_l) to use for final implementation. We aim to quickly eliminate design points that are either infeasible (due to constraints on cost, accuracy, or schedulability) or inferior to other points in terms of energy consumption.

IV. Proposed Approach for DSE

In this letter, our focus is to quickly eliminate the infeasible design points so that expensive evaluation techniques like simulations are invoked less. We define and use the following rules in Algorithm 1 for pruning the design space.

1) R1 – Accuracy: The system must support a minimum required accuracy level as determined by its use-case. All DNNs may not be able to satisfy the accuracy requirement for all applications. Rule R1 eliminates all application ($\mathbf{A}$) to DNN ($\mathbf{D}$) mappings that do not meet the accuracy requirement.

2) R2 – Area: Since one application uses only one DPU at a time, a maximum of n DPUs can be useful for the system. Within these choices, we eliminate all configurations that require more resources than available on the FPGA chip.

3) R3 – Individual utilization: Our applications are periodic and repeat themselves with a defined period (p). The temporal utilization of a DPU D_k by a DNN N^j executing application A_i (with period p_i) refers to the fraction of time for which the DPU is utilized by the application and defined as: $(u_k^j)_i = \frac{t_k^j}{p_i}$, where t_k^j is the runtime of N^j on D_k. For a mapping to be valid, the execution time should be less than the period. Hence, we eliminate all DNN to DPU mappings where $(u_k^j)_i > 1$. Therefore, certain smaller sized DPUs might not be valid choices for an application having smaller period.

4) R4 – Group utilization: Even though individual utilization of a DPU by an application might be less than 1 as per rule R3, multiple applications might have chosen the same DPU type. If sufficient instances of that DPU type are not available, schedulability would be violated. We consider multiple instances of same DPU type together as a group and check schedulability for the group. Assuming DPUs of type D_k have Cnt_k number of instances in a particular configuration, we eliminate all points for which the total utilization ($\sum_i (u_k^j)_i$) of DPU type D_k exceeds its count Cnt_k. Such a grouping avoids evaluating various symmetric mapping of tasks on same sized DPUs.

5) R5 – Platform configurations: We performed various measurements on ZCU102 board [16] (details in Section V-A) and derive two key designer insights. First, as shown in Fig. 1, the energy consumption for executing any DNN increases with decreasing size of DPU (or remains almost same as in smaller DNNs like mobilenet_v2 and squeezenet). Fig. 2 shows the measured increase in execution time and Fig. 3 shows the reduction in power averaged across DNNs (measured on board as well as reported from synthesis reports) and reduction in FPGA resources. The LUTs and registers do not decrease much with smaller DPUs and the area (and power) reduction achieved with a smaller DPU is less when compared to the increase in the execution time with use of smaller DPU. This explains the energy behavior shown in Fig. 1. Secondly, as shown in Fig. 4, running multiple DNNs back to back on a single DPU consumes more energy than concurrent execution on multiple DPUs of same type. This is because the power consumption of common resources (clock, bus, memory controller, and processing system) do not increase in proportion with multiple instances. The maximum increase in power consumption is 1.4× and 1.8× for 2 DPUs and 3 DPUs, respectively (Fig. 4). On average across DNNs, the total runtime for processing 1000 images reduces to 0.54× (with 2 DPUs) and 0.40× (with 3 DPUs) of the runtime for a single thread.

Across different Zynq chips, the detailed power for DPUs reported by Vivado tool showed similar trend as Fig. 3. Due to a fixed micro-architecture, a given DPU consumes same cycles across FPGA chips and hence, the runtime trend follows Fig. 2. Therefore, energy would behave similar to Fig. 1 and Fig. 4.

Using these observations about the energy consumption, we define the following pruning rule R5 for DPU based systems – eliminate all platform configurations that have the same number of DPUs, which are also smaller or same sized as previously evaluated configurations. The eliminated configurations cannot be Pareto points as they will consume higher energy and cannot execute faster than configurations that use larger or more number of DPUs.

Algorithm 1 shows the pseudocode for various rules, elaborating rule R5 in lines 3–26. The procedure starts with the configuration having larger sized DPUs and gradually replaces them with smaller sized DPUs while adding more instances. Lines 7–11 generate the first configuration containing maximum instances of largest DPU that can fit in the FPGA,

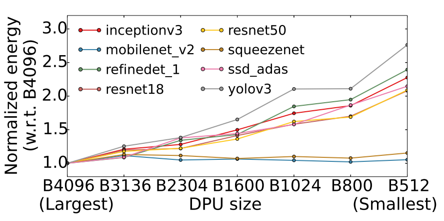

Fig. 1. Energy consumption for different DPU sizes (shown for a few DNNs for brevity)

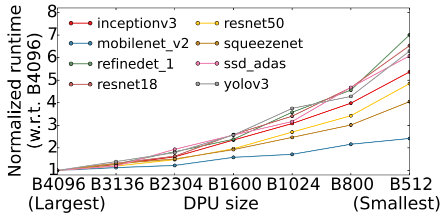

Fig. 2. Execution time for different DPU sizes for various DNNs

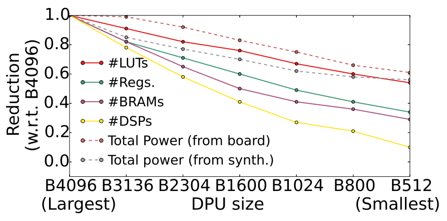

Fig. 3. Reduction in resource count and total power of FPGA chip for different DPU sizes

subsequently moving to the next largest DPU and so on. In the while loop from lines 12–25, the DPU at the last index in the first configuration is replaced with next smaller sized DPU (line 14). If such replacement allows adding another DPU, the configuration could be a Pareto point and is considered for evaluation (lines 16–19). If the last DPU is already the smallest DPU, it is removed and the same procedure is repeated with DPUs in previous index (lines 20–25). Valid configurations are further checked in line 13 for various mappings using other pruning rules indicated in the algorithm (lines 27–34).

We use an example of two applications (A_1 and A_2) to illustrate various pruning rules. There are 3 DNNs (N_1, N_2, N_3) with accuracy of A_1:(80, 0, 65) and A_2:(70, 68, 50). A_1 and A_2 should support an accuracy of 60 and period of 50 ms and 80 ms, respectively. There are 3 DPU types (D_1, D_2, D_3), s.t. $D_3 > D_2 > D_1$ with execution time (in ms) of N_1:(70, 50, 30), N_2:(62, 45, 25), and N_3:(60, 42, 20). Let us consider only one FPGA which can fit either (D_3+D_1) or (D_2+D_2) in it. Configurations like (D_3+D_3) or (D_3+D_2) are eliminated as they use larger DPUs and cannot fit on FPGA (rule R2). Since we can fit (D_3+D_1) and (D_2+D_2) on the FPGA, we eliminate configurations like (D_2+D_1), (D_1+D_1), (D_3), (D_2), (D_1) as they use smaller DPUs and would consume higher energy (rule R5). A_1 to N_2 and A_2 to N_3 mappings are eliminated in line 28 of Algorithm 1 due to accuracy requirement of 60 (rule R1). Line 30 will eliminate use of D_1 for A_1 due to runtime being more than its period (rule R3). For (D_2+D_2) configuration, we treat both D_2 as equivalent and line 30 checks the total utilization of D_2 to be less than 2 (rule R4). Overall, we were able to eliminate many design choices.

Algorithm 1: Overall DSE flow

```
1  for each FPGA type $F_l$ in F do
2      generateAndEvalConfig($F_l$)
3  Procedure generateAndEvalConfig($F_l$)
4      $availArea \leftarrow areaTotal_l$
5      $ind \leftarrow 0$
6      $C$ is an empty vector // chosen DPUs
7      for $k$ in $m..1$ do // generate first config.
8          $count \leftarrow \lfloor \frac{availArea}{area_k} \rfloor$
9          $C$.push($k$) for $count$ number of times
10         $availArea \leftarrow availArea - (area_k * count)$
11         $ind \leftarrow ind + count$
12     while true do
13         evalConfig($C$) // evaluate DPU config. C
14         $C[ind] = C[ind] - 1$ // Reduce DPU size
15         update $availArea$
16         if space gets created for another DPU $d$ then
17             $C$.push($d$) // This config. increases
                   concurrency and should be evaluated
18             update $availArea$
19             $ind \leftarrow ind + 1$
20         if $C[ind] = 0$ then // smallest DPU reached
21             if $ind$=0 then // all indices are covered
22                 return
23             else
24                 $C$.delete($ind$) // Delete the entry
25                 $ind \leftarrow ind - 1$ // Move to prev. index
26     return
27 Procedure evalConfig($C$) // C is DPU config.
28     for each A to N mapping do // use rule R1
29         for each A to C mapping do // use R4
30             if design point is infeasible then // use R3,R4
31                 continue
32             else
33                 Evaluate (simulate) the design point.
34     return
```

measure the execution time, power consumption, and energy consumption for each DNN, averaged over 1000 executions of the DNN. The accuracy for each application when mapped to a particular DNN is obtained from prior works [7], but standard measurements could also be performed if desired. The resource requirement for each DPU has been obtained from synthesis reports. The total available resources on a FPGA chip is obtained from corresponding datasheet. The cost for each FPGA type is obtained from the same vendor (DigiKey).

V. Evaluation and Results

A. Experimental Setup

1) Characterization methodology: We implement various hardware platforms containing multiple instances of different DPU types on a Xilinx Zynq Ultrascale+ based ZCU102 board [16] and execute different DNNs. The DPUs considered are: B512, B800, B1024, B1600, B2304, B3136, B4096 (the suffix number indicates peak execution rate in Giga-operations per second) [3]. We execute 15 standard classification and detection type DNNs on these DPUs: inception (v1, v2, v3), mobilenet_v2, squeezenet, resnet (50 and 18 layers), yolov3, refinedet (v1, v2, v3), ssd_(adas, pedestrian, traffic, mobilenet) [3]. We also create multiple threads (1, 2, and 3) of these DNNs to use multiple DPUs concurrently.

ZCU102 allows monitoring the power/current consumption through software using sensors on the supply rails. We measure the execution time, power consumption, and energy consumption for each DNN, averaged over 1000 executions of the DNN.

2) Applications (**A**)*:* We consider a driver assistance system that executes three different applications using camera images – A_1:person detection, A_2:vehicle detection, and A_3:road

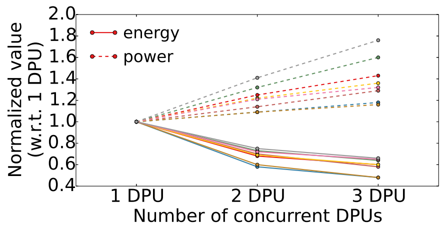

Fig. 4. Energy and power consumption for different number of DPUs (B4096) for various DNNs

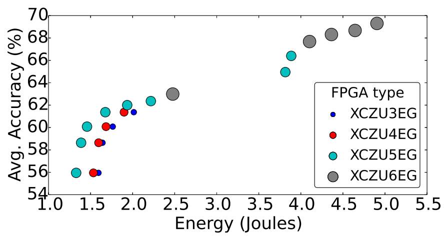

Fig. 5. Pareto points obtained after exploration. Size of points represent FPGA cost and size.

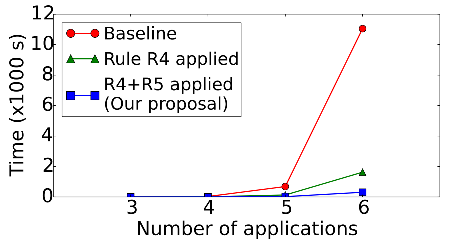

Fig. 6. Exploration time for different application sizes with different pruning approaches

TABLE I
DNN MAPPING AND PERIOD FOR DIFFERENT APPLICATIONS

Appln.	DNN and accuracy	Period (FPS)
A_1	refinedet_1: 67.47%, refinedet_2: 64.50%, refinedet_3: 60.65%	33.33 ms (30)
A_2	resnet50: 91.30%, inceptionv1: 89.41%, mobilenet_v2: 85.07%, squeezenet: 77.01%	33.33 ms (30)
A_3	yolov3: 49.14%, ssd_mobilenetv2: 30.19%	100.00 ms (10)

sign detection. Their accuracy (taken from Xilinx AI model kit [7]) and period for different DNNs are shown in Table I. The system provides useful information to a driver upfront. We consider the following design choices for DSE.

- **DNNs (N)**: DNN choices are shown in Table I.
- **DPUs (D)**: B512, B800, B1024, B1600, B2304, B3136, B4096. Suffix indicates peak execution rate in GOPs [3].
- **FPGA types (F)**: XCZU2EG, XCZU3EG, XCZU4EG, XCZU5EG, XCZU6EG, XCZU7EG, XCZU9EG, XCZU11EG, XCZU15EG. These are listed in increasing order of their cost and available resources.

B. Results

We compare our pruning strategy against a baseline strategy which implements rules R1, R2, and R3 and explores all platform configurations and all DNN to DPU mappings. We show improvements in two steps – (i) Grouping similar type of DPUs together to evaluate feasibility (rule R4) and (ii) Additionally, eliminating platform configurations based on designer insights (rule R4+R5).

We observe significant reduction in the number of design points evaluated and considered for simulation. While the baseline strategy evaluates 285288 design points and considers 25181 points for simulation, our proposed strategy (rule R4+R5) evaluates only 5376 design points (53× reduction) and considers 894 design points for simulation (28× reduction). This results in a runtime improvement of about 23×.

Fig. 5 shows that a bigger and more expensive FPGA (e.g., XCZU6EG) can support higher accuracy than smaller ones (e.g., XCZU3EG). Therefore, exploring various FPGA chips during DSE presents many new trade-offs in accuracy and cost compared to considering only one fixed FPGA chip, which to our knowledge has never been considered in any prior works. Very small FPGAs (e.g., XCZU2EG) get eliminated for not meeting periodicity whereas big FPGAs (e.g., XCZU7EG onwards) do not show up as Pareto points as additional resources offered by them are not needed for the system.

Next, we studied the quality of results obtained through different DSE strategies. Our approach eliminates the points which are definitely infeasible or inferior based on the insights obtained through experimentation. We examined that the design points eliminated due to rules R4 or R5 were non-Pareto points, which would have anyway been eliminated after simulation. Our strategy was able to prune many such design points early and thereby, reduced the exploration time.

We also studied scalability of the proposed approach for a larger number of applications (Fig. 6). Compared to the baseline, rule R4 and rules R4+R5 result in 6.8× and 36× lower exploration time, respectively for 6 applications.

VI. Conclusion and Future Work

We motivated the need for early stage DSE for systems with multiple DNN based applications. We obtained various insights from energy and runtime behavior for a commercial DNN accelerator (Xilinx DPU). Using these insights, we formulated a DSE strategy and deployed it to explore choice of FPGA types, DPU sizes, and application accuracy for a driver assistance system. We obtained 28× reduction in number of design points to be simulated and 23× improvement in pruning time, which nicely scales with higher number of applications.

In future, we plan to develop an energy estimation model to further reduce the number of simulations.

References

[1] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerators," *ACM TRETS*, 2019.
[2] S. I. Venieris and C. Bouganis, "f-CNNx: A toolflow for mapping multiple convolutional neural networks on FPGAs," in *FPL*, 2018.
[3] "DPU for CNN v3.0," 2019. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_0/pg338-dpu.pdf
[4] K. Guo *et al.*, "Angel-Eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE TCAD*, 2018.
[5] J. Peng *et al.*, "Multi-task ADAS system on FPGA," in *AICAS*, 2019.
[6] R. Kedia *et al.*, "MAVI: Mobility assistant for visually impaired with optional use of local and cloud resources," in *VLSID*, 2019.
[7] "Xilinx AI Model Zoo," August 2019. [Online]. Available: https://github.com/Xilinx/AI-Model-Zoo
[8] S. Baruah, "Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms," in *RTSS*, 2004.
[9] H. S. Chwa, J. Seo, J. Lee, and I. Shin, "Optimal real-time scheduling on two-type heterogeneous multicore platforms," in *RTSS*, 2015.
[10] S. Moulik, R. Devaraj, and A. Sarkar, "HEART: A heterogeneous energy-aware real-time scheduler," in *VLSID*, 2019.
[11] W. Quan and A. D. Pimentel, "A hybrid task mapping algorithm for heterogeneous MPSoCs," *ACM TECS*, 2015.
[12] P. van Stralen and A. Pimentel, "Scenario-based design space exploration of MPSoCs," in *ICCD*, 2010.
[13] G. Führ *et al.*, "Automatic energy-minimized HW/SW partitioning for FPGA-accelerated MPSoCs," *IEEE ESL*, 2019.
[14] S. Li *et al.*, "System level synthesis of hardware for DSP applications using pre-characterized function implementations," in *CODES*, 2013.
[15] W. Jiang *et al.*, "Accuracy vs. efficiency: Achieving both through FPGA-implementation aware neural architecture search," in *DAC*, 2019.
[16] "Zynq UltraScale+ MPSoC ZCU102 evaluation kit." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html