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 $28 \times$ in the number of design points to be simulated and $23 \times$ 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 $28 \times$ reduction in the number of invocations of the simulator/estimation function and an overall speedup of $23 \times$ 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+ [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

$A = \{A_1, A_2, \ldots, A_n\}$ is a set of $n$ periodic applications repeating at a defined interval known as period. Each application $A_i = \langle a_i^1, a_i^2, \ldots, a_i^n, p_i\rangle$, where $p_i$ represents the period.
of $A_i$ and $a^2_i$ represents the accuracy of application $A_i$ when
mapped to network (DNN) $N^j$. If application $A_i$ cannot use
$N^j$, then $a^2_i = 0$.
$N = \{N^1, N^2, \ldots, N^s\}$ is a set of $s$ different networks.
Each network $N^j$ is characterized as: $N^j = \langle t^j_1, t^j_2, \ldots, t^j_m \rangle$
where $t^j_k$ 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].
$D$ is an ordered set $\langle \{D_1, D_2, \ldots, D_m\} \rangle$ 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 $D_k$. $D$ 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.
$F = \{F_1, F_2, \ldots, F_r\}$ is a set of $r$ FPGA chips. Each FPGA
chip $F_i = \langle areaTotal_i, cost_i, f_i \rangle$, where $areaTotal_i$
represents the total resources (in terms of LUT, BRAM, and DSP),
$cost_i$ represents the price, and $f_i$ represents the normalized
frequency of the FPGA w.r.t. a reference FPGA (ZCU102).
$f_i$ is used for scaling the execution time for different FPGA
(currently only Zynq devices support Xilinx DPU).

B. Problem Statement
The design space being explored in this letter consists of various
application (A) to DNN (N) and DNN (N) to DPU (D) mappings.
It considers count of each DPU type ($D_k$) to be used, mapping
of applications on allocated DPUs, and the FPGA chip ($F_i$) 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 (A) to DNN
(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 that the
DPU is utilized by the application and defined as: $(u^j_k)_i = \frac{t^j_k}{p_i}$,
where $t^j_k$ 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 $(u^j_k)_i > 1$.
Therefore, certain smaller sized DPUs might not be valid
choices for an application having smaller period.

4) $R4$ – Group utilization: Even though individual utili-
zation 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 ($\sum_i (u^j_k)_i$) 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 $1.4 \times$ and $1.8 \times$ for 2 DPUs and 3
DPUs, respectively (Fig. 4). On average across DNNs, the total
runtime for processing 1000 images reduces to 0.54$\times$ (with 2
DPUs) and 0.40$\times$ (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, elab-
orating 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 max-
inimum instances of largest DPU that can fit in the FPGA,
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 (27–34).

We use an example of two applications (A1 and A2) to
discuss various pruning rules. There are 3 DNNs (N1, N2, N3) with accuracy of A1:(80, 0, 65) and A2:(70, 68, 50). A1
and A2 should support an accuracy of 60 and period of 50 ms
and 80 ms, respectively. There are 3 DPU types (D1, D2, D3),
s.t. D3 > D2 > D1 with execution time (in ms) of N1:(70, 50,
30), N2:(62, 45, 25), and N3:(60, 42, 20). Let us consider only
one FPGA which can fit either (D1+D1) or (D2+D2) in it.
Configurations like (D2+D2) or (D3+D3) are eliminated as
they use larger DPUs and cannot fit on FPGA (rule R2). Since
we can fit (D3+D1) and (D2+D2) on the FPGA, we eliminate
configurations like (D3+D1), (D1+D1), (D2), (D1) as
they use smaller DPUs and would consume higher energy (rule
R5). A1 to N2 and A2 to N3 mappings are eliminated in line
28 of Algorithm 1 due to accuracy requirement of 60
(rule R1). Line 30 will eliminate use of D1 for A1 due to
timeout being more than its period (rule R3). For (D2+D2)
configuration, we treat both D2 as equivalent and line 30
checks the total utilization of D2 to be less than 2 (rule R4).
Overall, we were able to eliminate many design choices.

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. 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).

2) Applications (A): We consider a driver assistance sys-
tem that executes three different applications using camera
images – A1: person detection, A2: vehicle detection, and A3: road
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


