Leveraging PVT-Margins in Design Space Exploration for FPGA-based CNN Accelerators

Weina Lu*†, Wenyan Lu*†, Jing Ye*, Yu Hu*†, Xiaowei Li*†
*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
†Graduate University of Chinese Academy of Sciences
{luweina, luwenyan, yejing, huyu, lxw}@ict.ac.cn

Abstract—**The performance of an FPGA based CNN accelerator is determined by both parallelism and frequency, however, most prior works optimize the parallelism in the RTL design and resolve the frequency after the synthesis. This paper presents a design space exploration method for the pipeline implementation of the deep CNN models, which concurrently optimizes parallelism and frequency to achieve a comprehensive optimization on throughput. In addition to the quantitative modeling on parallelism, the maximum achievable system frequency under various parallelism is explored to leverage the PVT-margins in real-life scenarios and is adopted to guide the design space exploration for further performance boost. A case study of the AlexNet model is implemented using the proposed method on the Altera DE5a-Net board. The experimental results demonstrate that our method can achieve the throughput up to 906.25GOP/s, which gains $1.39\times$ improvement compared to state-of-the-art RTL optimization methods.**

I. Introduction

In recent years, field programmable gate arrays (FPGAs) exhibit superior energy efficiency in the high-performance implementation of deep convolutional neural networks (CNNs). The throughput is computed by the system frequency and the parallelism which is measured by the number of operations per cycle [1].

The system frequency and the parallelism have a mutual influence with each other, due to the constraints of resource and bandwidth of the hardware. However, recent works on RTL-level accelerator designs mainly focus on the enhancement of parallelism but rarely taking the frequency into consideration [1]–[6]. The implementation-level optimization methods optimize both of parallelism and frequency by enhancing the parallelism in the RTL design and improving the frequency in the placement or routing [7], [8]. Since the independent optimizations take little account of the mutual influence between the two aspects, the throughput of these methods may be optimized but may not be maximized. The work in [9] optimizes the throughput with concurrent consideration of parallelism and frequency. But most of the models are established with empiricism, which decreases the commonality for application. In addition, the kernel frequency in the model is derived from the synthesis data, which is extremely conservative.

As the fabricating technology advancements, process variation becomes more prominent [10]. Commercial design tools usually adopt the worst-case process, voltage and temperature (PVT) as one of the guidance for placement and routing to ensure the reliability of the circuit. However, the worst-case PVT casts a large deviation to the reality and thus makes the frequency reported by the commercial tool much lower than the maximum achievable frequency in real-life scenarios. It is reported that the performance can be improved by 30.70% with leveraging real-life PVT margins on a 60nm FPGA [11]. Traditional FPGA-based overclocking techniques usually delve the frequency enhancement with general circuits without considering the scenarios of high resource or bandwidth requirements [12]–[14], which are less practical for throughput-oriented applications such as deep CNN accelerators. Therefore, in this paper, we come up with a design space exploration method for the FPGA-based CNN accelerators which leverages the real-life maximum achievable frequency for efficient performance boost.

The key contributions of our work are listed as follows.

1) The maximum achievable frequency under various parallelism is explored and analytically modeled for the accelerator to leverage PVT-margins in real-life scenarios.
2) The relationships among frequency, parallelism and their influence factors are analyzed and modeled in details.
3) A design space exploration method with concurrently optimizing the frequency and parallelism is proposed for the deep CNN models for comprehensive optimization on the throughput.
4) A case study of the AlexNet model is implemented using our method on Altera DE5a-Net board and achieves $1.39\times$ improvement on the throughput.

II. Framework Overview

In the large-scale CNNs, since the size of feature maps and weights are different among layers, we apply different parallel strategies for different layers and adopt the pipeline structure where each layer is treated as a pipeline stage for implementation. The overall framework is proposed at RTL-level design. As shown in Figure 1, it is mainly composed of a CNN accelerator, a clock manager and a temperature monitor.

The clock manager and the temperature monitor are used for the exploration of real-life maximum achievable frequency. The clock manager is implemented by a fractional phase-locked loop (PLL) to provide different frequencies for the system. The temperature monitor is connected to a temperature diode on the FPGA to measure the actual on-chip temperature when running the accelerator. The exploration procedure is

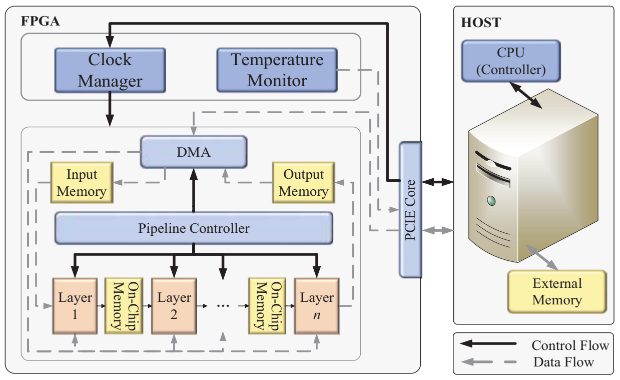

Fig. 1: The Proposed Framework

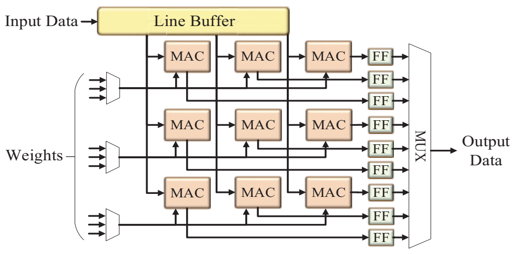

Fig. 2: The Structure of 2-D PE

flexible and requires little logic elements on the FPGA because both of the components can be implemented with the dedicated hard IP [15], [16] and the controller of the two components is implemented as software running on the host computer.

The multi-stage pipeline and the pipeline controller are the components for the implementation of the deep CNN accelerator. The number of pipeline stages is the same as the number of layers in the CNN model. The on-chip memories are connected between layers to store the intermediate results. All the weights are read from external memory by PCIE and controlled by DMA.

The structure in a pipeline stage is implemented with the typical structure which implements the pooling, ReLU or LRN function combined with the convolution, as in [3]. The 2-D PE is adopted to implement the convolution kernel, as shown in Figure 2. The MACs are arranged in a $ceil(\frac{W}{S}) \times ceil(\frac{W}{S})$ array with the input of each column separated by $S-1$ buffers. Here, W is the width of the kernel and S is the stride. The convolution for an input map can be completed with $[ceil(\frac{W}{S})]^2 \cdot R_{in} \cdot C_{in}$ operations with the presented PE structure where R_{in} and C_{in} are the number of rows and columns in an input feature map. The FC layers are implemented by matrix multiplication. Since they are memory-intensive, we adopt the batch-based computing for the FC layers.

III. Design Space Exploration

The optimization objective in our work is to maximize the throughput of the CNN accelerator in real-life scenarios. The throughput can be computed with the product of system frequency and the number of operations per cycle.

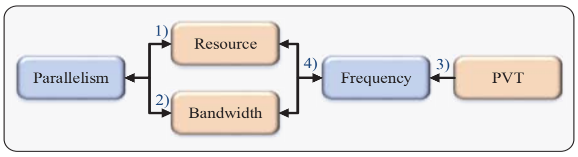

Fig. 3: Relationships between Frequency and Parallelism

The implementation of a pipeline expects the execution time of each pipeline stage to be approximately the same. We denote the number of clock cycles for each pipeline stage as N_{cycle}. Since the number of operations for each layer should be completed in N_{cycle}, the throughput of the CNN accelerator can be expressed as in Equation 1 where N_{in} and N_{out} are the number of input and output feature maps, respectively. f is the system frequency.

$$\begin{aligned} TP &= Ops_{cycle} \cdot f \\ &= \frac{\sum_{i=1}^{N_{conv}} R_{in}^i C_{in}^i N_{in}^i N_{out}^i [ceil(\frac{W^i}{S^i})]^2 + \sum_{i=1}^{N_{fc}} R_{in}^i R_{out}^i}{N_{cycle}} \cdot f \\ &= \frac{f}{N_{cycle}} (\sum_{i=1}^{N_{conv}} R_{in}^i C_{in}^i N_{in}^i N_{out}^i [ceil(\frac{W^i}{S^i})]^2 + \sum_{i=1}^{N_{fc}} R_{in}^i R_{out}^i) \end{aligned} \tag{1}$$

Since most of the parameters can be extracted from the CNN model except for N_{cycle} and f, our optimization objective is to maximize the $\frac{f}{N_{cycle}}$ under the constraints of the available $Resource$, $Bandwidth$ and the maximum achievable frequency F_{max}^{PVT} under the real-life PVT, as described below.

Objective: $Maximize(\frac{f}{N_{cycle}})$

Subject to

$$f, N_{cycle} \le Resource \tag{2}$$

$$f, N_{cycle} \le Bandwidth \tag{3}$$

$$f, N_{cycle} \le F_{max}^{PVT} \tag{4}$$

A. *In-Depth Analysis on the Frequency and Parallelism*

The relationships among frequency, parallelism and their influence factors are depicted in Figure 3. We firstly discuss each influence aspects in details and then model them to the constraints in the problem formulation.

1) Constraints of the Resource: The multiplication in the CNN model is implemented with on-chip DSPs. The number of required DSPs for a layer is computed by the division of the total number of multiplications of a pipeline stage and the N_{cycle}. The intermediate results between layers are stored in the on-chip memories. Thus the size of the required memory followed by a layer should be greater than the size of its output data. The number of required on-chip memories by a layer is computed by the division of the total required data size and the capacity of one BRAM.

Each type of resources required by the accelerator is preferred not greater than the availability of the FPGA. Therefore, the constraints of $Resource$ can be formulated by Equation 5, 6 and 7, where C_{BRAM} is the capacity of one BRAM. The memory requirements of FC layers are increased by batch size N to store the intermediate data for N input maps.

$$N_{cycle} \geq \frac{\sum_{i=1}^{N_{conv}}(\frac{W^i}{S^i})^2 R_{in}^i C_{in}^i N_{in}^i N_{out}^i + \sum_{i=1}^{N_{fc}} R_{in}^i R_{out}^i}{N_{DSP}^{total}} \tag{5}$$

$$N \leq \frac{N_{BRAM}^{total} - \sum_{i=1}^{N_{conv}} N_{BRAM}^i}{\sum_{j=1}^{N_{fc}} N_{BRAM}^j} \tag{6}$$

$$N_{BRAM}^i \geq \frac{R_{out}^i \cdot C_{out}^i \cdot N_{out}^i}{C_{BRAM}} \tag{7}$$

2) Constraints of Memory Bandwidth: The bandwidth required by a layer is computed by the product of the frequency and the number of data access with external memory per cycle. Since all the intermediate data are stored in the on-chip BRAMs and only weights are read from extern, the total required bandwidth is the sum of weight requirements of each layer. The required bandwidth is preferred not greater than but as close to the available bandwidth as possible for achieving high performance. Thus the constraint of $Bandwidth$ can be expressed as in Equation 8 where the bit width of one data is denoted as BW_{bit}. The batch based computing effectively increases the data reuse of weights for the FC layers and decrease their bandwidth requirements by $\frac{1}{N}$.

$$\frac{f}{N_{cycle}} \leq \frac{BW_{total}}{[\sum_{i=1}^{N_{conv}}(W^i)^2 N_{in}^i N_{out}^i + \frac{1}{N}\sum_{i=1}^{N_{fc}} R_{in}^i R_{out}^i]BW_{bit}} \tag{8}$$

3) Constraints of the PVT: As one of the divisors in the throughput, the frequency owns the same importance as parallelism. However, commercial design tools usually adopt the worst-case PVT for timing analysis, such as the temperature deviation demonstrated in [17], [18]. Therefore, we explored the maximum achievable frequency in real-life scenarios and adopted it as a guidance for design space exploration.

In the maximum achievable frequency exploration, a series of frequencies ranging from the tool-reported frequency to the maximum frequency that can be obtained by PLL with an increment of approximately 10MHz are provided by the clock manager. A set of input maps with known output are adopted as a test case and are input to the CNN accelerator recurrently during the exploration. The frequency is initially set as the tool-reported frequency and is increased step by step. Since the increase in frequency would disturb the on-chip thermal equilibrium, the computational results are checked after the on-chip temperature comes into stable. This guarantees the reliability of the circuits working on maximum achievable frequency in real-life scenarios as well. The maximum achievable frequency is worked out when the increase in frequency give rise to any error in the output results.

4) Relationship of Parallelism and Frequency: The parallelism exerts indirect influence on the system frequency.

A high resource utilization would give rise to routing congestion and decline the maximum achievable frequency. The resource requirements of the CNN accelerator increase with the improvement of parallelism, which indicates a degradation on the frequency. We model the explored maximum achievable frequency F_{max}^{PVT} with different N_{cycle} as a fitting function, as shown in Equation 9.

$$F_{max}^{PVT}(N_{cycle}) = f(N_{cycle}) \tag{9}$$

On the other hand, the bandwidth required by the parallelism would also limit the maximum frequency, which has been demonstrated in Equation 8. We denote the limitation of bandwidth on the frequency as $F_{max}^{bandwidth}$ which can be formulated as a function of N_{cycle} and batch size N, as shown in Equation 10.

$$\begin{aligned} &F_{max}^{bandwidth}(N_{cycle}, N) \\ &= \frac{BW_{total} \cdot N_{cycle}}{[\sum_{i=1}^{N_{conv}}(W^i)^2 N_{in}^i N_{out}^i + \frac{1}{N}\sum_{i=1}^{N_{fc}} R_{in}^i R_{out}^i]BW_{bit}} \end{aligned} \tag{10}$$

The maximum frequency F_{max} ought to simultaneously satisfy the above two aspects, as shown in Equation 11.

$$F_{max} = min\{F_{max}^{PVT}(N_{cycle}), F_{max}^{bandwidth}(N_{cycle}, N)\} \tag{11}$$

B. Problem Solving

The solving procedure for the formulated problem is as follows. First, the constraint in Equation 5 is worked out for a given CNN model and a target FPGA. Then under this feasible region, the maximum achievable frequency F_{max}^{PVT} under different N_{cycle} is explored and is modeled by a fitting function. Since the constraints in Equation 5-11 restrict the feasible region of f and N_{cycle} quite small, the optimal f and N_{cycle} can be worked out by an exhaustive algorithm.

The batch size N are dynamically adjusted along with the exhaustion procedure to maximize the bandwidth utilization by relaxing the restriction of bandwidth on the maximum achievable frequency. It should meanwhile satisfy the BRAM constraint in Equation 6. The actual parallel strategy which is represented by the number of PE components P_i in the ith layer is adjusted to guarantee the actual number of clock cycles in each pipeline stage not exceeding the N_{cycle}.

IV. Experimental Results

The proposed method is demonstrated by implementing the AlexNet with parameterized Verilog scripts on the Altera DE5a-Net board via *Quartus prime* 16.0. All the experiments are conducted with the ambient temperature of 30°C. The AlexNet is implemented with an 8-stage pipeline with 5 convolution stages and 3 FC stages. 16-bit fixed point data is proven to be accurate enough for computation [19] and thus is adopted for the weights, input feature maps and the intermediate results.

In the exploration of maximum achievable frequency, the F_{max}^{PVT} has an approximately consistent trend with the tool-reported frequency and achieves over 10% improvement in our experiments. A dozen of fitting functions such as the *Logistic* or *Lognormal* is provided for chosen. The inversely exponential decay function *ExpDecay* achieves the highest

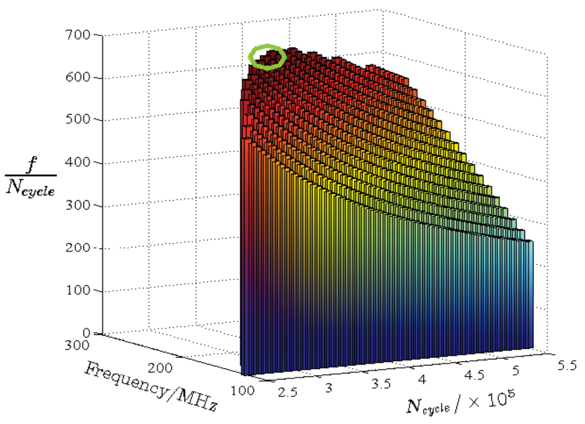

Fig. 4: The Objective $\frac{f}{N_{cycle}}$ under Various f and N_{cycle}

fitting precision which is over 99.8% and thus is adopted to express the relationship between the F_{max}^{PVT} and N_{cycle}.

The optimization objective, $\frac{f}{N_{cycle}}$, under various frequency f and N_{cycle} are depicted in Figure 4. The maximum $\frac{f}{N_{cycle}}$ achieves in the green circle which is 625 with $N_{cycle} = 3.6 \times 10^5$ and $N = 33$. And the parallel strategy for each layer is detailed in Table I.

TABLE I: Parallel Strategy for Each Layer

Layer	Conv1	Conv2	Conv3	Conv4	Conv5	FC6	FC7	FC8
P_i	48	32	48	36	32	128	64	16

To further demonstrate the effectiveness of our method, the original RTL optimization method [4] which takes the parallelism as the optimization objective is re-implemented as a baseline for comparison. The design report of the baseline and our method are listed in Table II.

TABLE II: Design Report of the Methods

Design Report	Resource Type			Frequency
	ALM	DSP	BRAM	(0.9V, 100°C)
The Baseline	67.52%	97.23%	49.71%	163MHz
The Proposed	55.34%	81.82%	71.66%	200MHz

The comparison between the two methods is illustrated in Table III. Even though the baseline has higher utilization on computational resources, the higher parallelism increases the data access with the external memory and seriously decreases the space for frequency enhancement. In consequence, the performance does not get any improvement even if the baseline is working at a higher frequency in the real-life scenario, because the computations have to wait for the weights loading. In contrast, our work can fully exploit the bandwidth and frequency and achieve a comprehensive optimization on performance. The maximum throughput of our method is up to 906.25GOP/s, which is 38.89% higher than the baseline. This further demonstrates the importance of using the real-life maximum achievable frequency for design space exploration.

TABLE III: Comparison of the Proposed Method and Baseline

Techniques	N_{cycle}	F_{max}^{PVT} (62°C,1.8V)	$F_{max}^{bandwidth}$	Actual Throughput	
The Baseline	3×10^5	180MHz	**135MHz**	450img/s	↑ ×**1.39**
The Proposed	3.6×10^5	**225MHz**	231MHz	625img/s	

V. Conclusion and Future Works

In this paper, we propose a design space exploration method, which concurrently optimizes the parallelism and the system frequency to achieve a comprehensive maximization on the throughput. Future works would apply the proposed method to various CNN model for verification and adopt appropriate thermal management for online timing variation to avoid the timing violations on the FPGA.

VI. Acknowledgments

This work is supported in part by National Natural Science Foundation of China under grant No. 61274030, 61532017, 61376043, 61521092. Yu Hu is the corresponding author.

References

[1] C Zhang, D Wu, J Sun, et al. Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster. In *ISLPED*, 2016.
[2] C Zhang, P Li, G Sun, et al. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In *FPGA*, 2015.
[3] J Qiu, J Wang, S Yao, et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In *FPGA*, 2016.
[4] Li H, Fan X, Jiao L, et al. A high performance FPGA-based Accelerator for Large-scale Convolutional Neural Networks. In *FPL*, 2016.
[5] Y Ma, Y Cao, S Vrudhula, et al. Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks. In *FPGA*, 2017.
[6] Y. Wang, J. Xu, Y. Han, et al. DeepBurning: Automatic Generation of FPGA-based Learning Accelerators for the Neural Network Family. In *DAC*, 2016.
[7] J Zhang and J Li. Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network. In *FPGA*, 2017.
[8] U Aydonat, S OConnell, D Capalija, et al. An OpenCL Deep Learning Accelerator on Arria 10. In *FPGA*, 2017.
[9] N Suda, V Chandra, G Dasika, et al. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks. In *FPGA*, 2016.
[10] S Mittal. A Survey of Architectural Techniques for Managing Process Variation. In *ACM CSUR*, 2016.
[11] J M. Levine, E Stott, and P Y.K. Cheung. Dynamic Voltage & Frequency Scaling with Online Slack Measurement. In *FPGA*, 2014.
[12] R. P. Duarte and C.-S. Bouganis. High-level Linear Projection Circuit Design Optimization Framework for FPGAs under Over-clocking. In *FPL*, 2012.
[13] K. Shi, D. Boland, and G. A. Constantinides. Accuracy-Performance Tradeoffs on an FPGA through Overclocking. In *FCCM*, 2013.
[14] K Maragos, G Lentaris, and D Soudris. Application Performance Improvement by Exploiting Process Variability on FPGA Devices. In *DATE*, 2017.
[15] Intel Altera. Altera I/O Phase-Locked Loop (Altera IOPLL) IP Core User Guide. *https://www.altera.com*, 2016.
[16] Intel Altera. Altera Temperature Sensor IP Core User Guide. *https://www.altera.com*, 2015.
[17] W Lu, Y Hu, J Ye, et al. TeSHoP: A Temperature Sensing based Hotspot-Driven Placement Technique for FPGAs. In *FPL*, 2016.
[18] W Lu, Y Hu, J Ye, et al. Going Cooler with TeSHoP: A Temperature Sensing based Hotspot-Driven Placement Technique for FPGAs. In *IEEE TVLSI*, 2017.
[19] M. Courbariaux and Y. Bengio. Low Precision Storage for Deep Learning. In *ICLR*, 2015.