XPPE: Cross-Platform Performance Estimation of Hardware Accelerators Using Machine Learning

Hosein Mohammadi Makrani, Hossein Sayadi, Tinoosh Mohsenin,
Setareh rafatirad, Avesta Sasan, Houman Homayoun
George Mason University
Fairfax, VA 22030
{hmohamm8,hsayadi,tmohseni,srafatir,asasan,hhomyou}@gmu.edu

ABSTRACT

The increasing heterogeneity in the applications to be processed ceased ASICs to exist as the most efficient processing platform. Hybrid processing platforms such as CPU+FPGA are emerging as powerful processing platforms to support an efficient processing for a diverse range of applications. Hardware/Software co-design enabled designers to take advantage of these new hybrid platforms such as Zynq. However, dividing an application into two parts that one part runs on CPU and the other part is converted to a hardware accelerator implemented on FPGA, is making the platform selection difficult for the developers as there is a significant variation in the application's performance achieved on different platforms. Developers are required to fully implement the design on each platform to have an estimation of the performance. This process is tedious when the number of available platforms is large. To address such challenge, in this work we propose XPPE, a neural network based cross-platform performance estimation. XPPE utilizes the resource utilization of an application on a specific FPGA to estimate the performance on other FPGAs. The proposed estimation is performed for a wide range of applications and evaluated against a vast set of platforms. Moreover, XPPE enables developers to explore the design space without requiring to fully implement and map the application. Our evaluation results show that the correlation between the estimated speed up using XPPE and actual speedup of applications on a Hybrid platform over an ARM processor is more than 0.98.

CCS CONCEPTS

• **Computer systems organization → Reconfigurable computing**;

KEYWORDS

Design space exploration, performance estimation, machine learning, accelerator

ASPDAC '19, January 21–24, 2019, Tokyo, Japan

ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3288756

ACM Reference Format:
Hosein Mohammadi Makrani, Hossein Sayadi, Tinoosh Mohsenin,, Setareh rafatirad, Avesta Sasan, Houman Homayoun. 2019. XPPE: Cross-Platform Performance Estimation of Hardware Accelerators Using Machine Learning. In *24th Asia and South Pacific Design Automation Conference (ASPDAC '19), January 21–24, 2019, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3287624.3288756

1 INTRODUCTION

The end of Dennard Scaling era and the thrive to achieve high performance led to evolution of new computer architecture designs[10]. ASICs persist to be no more the best executing hardware platform due to the design complexity, involved costs and the time-to-market challenges [1, 19].

New hybrid platforms such as CPU+FPGA systems are emerging as the potential solution, despite the fact that FPGAs are nearly one order magnitude slower than the specialized ASICs [9]. FPGAs enjoy other benefits such as on-the-fly programmability, reconfigurability, energy-efficiency, and the development of hardware/software co-design platforms[2, 18]. This also facilitates engineers to perform hardware design without requiring deeper insights into hardware design [7].

As there exists a performance gap among the implementation of an application on various FPGA devices (1×-1000×), it is important for the developers to determine which of the FPGA platforms to choose. This challenge is further complicated with the availability of a large number of FPGA boards. To yield the high-performance by running appropriate applications on FPGAs, it is important to perform the design space exploration (DSE) [7] using timing analysis provided by CAD tools. After that, the designer can decide to implement an application on the suitable platform.

There are challenges associated with the static timing analysis of digital systems designs [4]: The latest version of CAD tools provided by Xilinx (Vivado), does not have the capability to report the maximum frequency achievable for the corresponding code. The user must request a target frequency, and the tool reports either a "pass" or "fail" for its attempt to achieve this goal [5]. While there are 25 optimization strategies predefined in the tool, applying them sequentially, is extremely tedious and time consuming. Therefore, the estimation of the achievable performance improvements will aid the designer to choose faster and wisely among different FPGA devices.

The performance estimation for choosing the platform to run the application faces the challenges from the vast heterogeneity of existing type of applications and the FPGA devices. Most of the existing works predict performance based on the application characteristics

and depending on the resource consumed [7–9, 16], but limited to specific FPGAs. Moreover, these works lack a detailed comparison of achieved performance benefits to the processor subsystems.

In this work, we classify FPGAs into three categories: low-end, medium, and high-end FPGAs based on the available resources on the FPGA. Furthermore, it has been shown that the performance of an application can be enhanced by varying the parameters of FPGA such as frequency, memory bandwidth [11, 13], and so on [27]. As such, we explore the impact of different parameters and the class of the FPGA on the performance of an application.

Based on the resource utilization of an application across different FPGA's classes and the achieved performance (speedup), we build a machine learning based cross platform performance estimation (XPPE) tool. XPPE uses the resource utilization of an application reporeted by Xilinx HLS tool and predicts the sped up of that applications on different platforms. Our evaluation shows the predicted speed up and the actual speedup over ARM Cortex A-9 MPCore processor for different wide range of applications has a correlation of R=0.98.

The main contributions of this work are:

- Design space exploration of different classes of FPGAs and determine the parameters that have significant impact on the performance w.r.t the type of application.
- Provide a cross-platform performance estimation tool (XPPE) based on the machine learning model to determine the performance improvement (acceleration) over a general purpose processor (ARM Cortex A-9 MPcore in our experiments).

2 EXPERIMENTAL SETUP

The experimental setup for building a machine learning model, and the methodology to determine the important parameters of FPGAs for performance boosting is presented in this section. We first require a dataset for training the model. Hence, we present our benchmarks, FPGA devices, and the methodology of performing our experiments to collect the required data.

2.1 Studied applications

In order to model the performance of applications, a dataset was generated from popular HLS benchmark suits to make sure that the diversity of benchmarks is comprehensive. The selected benchmarks are from Machsuit [17], S2CBench [23], CHStone [6], Rosetta [29]. To increase the diversity and the size of dataset, we also used a collection of 10 different image processing kernels from Xilinx xfOpenCV. We totally covered 70 benchmarks which include a wide range of domains from simple kernels to machine learning and real-time video processing that they reflect the latest application trends.

Each application was implemented and verified for functional correctness before analyzing across different FPGAs studied.

As a basis for comparison, all applications were run in a single-threaded manner on an ARM A9 processor with a 650 MHz CPU clock. The functions used for the software baseline were standard functions that were not optimized for FPGA. The software utilized for testing was the Xilinx SDSoC development suite, which uses Vivado and Vivado HLS version 2017.2. as its underlying software.

Table 1: FGPA devices' specification

Family	Device	Frequency (MHz)	Technology	DRAM interconnect speed	Class
Zynq	XC7Z010	125	28	1866	Low-end
	XC7Z020	200	28	1866	
	XC7Z045	200	28	1866	
Artix-7	XC7A50T	200	28	1066	
	XC7A100T	200	28	1066	
	XC7200T	200	28	1066	
Zynq Ultrascale+	XCZU9EG	300	16	2666	Medium
Virtex-7	XC7VX485T	300	28	1866	
	XC7VX690T	300	28	1866	
	XC7V2000T	400	28	1866	
Ultrascale+	XCVU5P	300	16	2666	
Zynq Ultrascale+	ZU19EG	300	16	2666	
UltraScale	XCVU065	300	20	2400	
Ultrascale+	XCVU3P	300	16	2666	
UltraScale	XCVU095	300	20	2400	High-end
	XCVU190	400	20	2400	
Ultrascale+	XCVU11P	400	16	2666	
UltraScale	XCVU125	400	20	2400	
	XCVU440	400	20	2400	
Ultrascale+	XCVU13P	400	16	2666	

2.2 Hardware platforms

For selecting FPGA devices, we targeted three different class of FPGAs such as Low-end, Medium-end, and High-end. We selected all of our devices from Xilin. 20 different FPGA technologies were tested to generate the dataset. Table 1 shows the FPGAs and their classes. Devices from three main Xilinx families were chosen: Zynq, Artix, and Virtex. Additionally, from the Zynq and Virtex family, devices from among the 28nm, 20nm (UltraSCALE), and 16nm (UltraSCALE+) were used. The FPGA devices are chosen based on a wide array of available resources and technologies across the spectrum of each family.

2.3 Performance measurement

We kept the hardware/software co-design relatively simple, with only the data transceiver and hardware accelerator located in the PL part. The source and destination data was transferred directly to the accelerator via an AXI-4 data bus. This method of data transport is considered directly into our timing model of hardware performance as well as into the device utilization. Other parameters to the accelerator functions (such as filter size specifications, tuning parameters, etc.) were implemented as standard data ports to the accelerator.

To measure the performance, each application was synthesized in HLS for its respective device. Timing information for the estimated hardware performance was gathered using the reported clock-cycle count of the maximum latency of the accelerator and plus data transceiver. In order to estimate the amount of time that was spent in hardware and software for the co-design of each application, applications are first evaluated on fully implementable SoCs from the Zynq family (Steps 4-5 in Figure 1). A performance analysis was executed on each application to determine the overall execution time of the full design. This is used for estimating the total execution time. Tough the tool provided a worst-case estimation, it provided a trend that could be used for total execution time. (It should be noted, however, the performance estimation by the tool had a tendency to

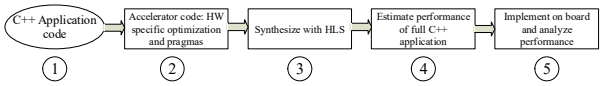

Figure 1: Experimental FPGA design flow for obtaining performance

Table 2: Average speed up of applications on 3 different FPGAs over Arm processor

Application Type	High-End (XCVU13P)	Medium-End (XCZU9EG)	Low-End (XC7Z010)
Machine Learning	26.27	19.02	5.51
Image/Video Processing	155.71	68.42	9.26
Cryptography	23.32	12.74	3.19
Mathematical	66.65	28.43	6.78

be more than 1.5-1.75× the execution time of an application tested on a fully implemented design.)

A ratio of the time spent in Programmable Logic (PL) vs Processor System (PS) was then calculated using the synthesis reports from HLS and the overall performance estimations. The calculated ratios for different Zynq boards varied only by about 1-2%, so these ratios could be used with decent confidence for the remainder of the technologies tested. These ratios were then applied to the hardware execution times for the FPGA technologies that were not available as SoC platforms. Table 2 presents achieved acceleration compared to ARM A9 processor for different types of applications on three FPGA devices. This table is important as it shows interesting trends that we will discuss it in section 4.

2.4 Data collection

To build a dataset, we first require to extract all possible features that we can get from HLS reports. HLS related features consist the inputs of our machine learning model. Table 3 shows the features that we can extract from HLS reports. As it is not wise to determine the importance of each feature in advance, we extract as many relevant features as possible (total 183 features). We do not use principal component analysis to reduce the dimensionality of features. Because if there is a correlation between an additional feature and the target variable, the model can learn it. If there is no correlation, the model learns to ignore. Therefore, more features are at least as good as a model with only a subset of the same features. However, too many features may lead to the over-fitting problem. Hence, we address this issue later in this paper. Moreover, the machine learning model will be trained to estimates performance speed up of application if implemented on various FPGA devices. For cross-validation of the accuracy of the machine learning model and according to the accepted practice in data analytics, we set the testing size a quarter of the size of the dataset.

3 XPPE

We introduce XPPE (cross-platform performance estimator), an automated performance prediction tool to estimate the speedup of an application when implemented as a hardware accelerator on FPGA devices. The estimation is based on the resource utilization report by HLS Vivado tool, available resources on target FPGA,

Table 3: Description of features

Category	Brief Description	Number of features
Performance	Requested clock period, estimated clock period by HLS, Uncertainty	3
Resources	Utilization and availability of LUT, FF, DSP, and BRAM	36
Logic and arithmetic operation	Bitwidth/resource statistics of operations	132
Memory	Number of memory words/banks/bits;	8
Multiplexer	Multiplexer input size/bitwidth	4

and application's characteristics. The promising part of XPPE is that it is enabled to estimate the speed up of accelerator on any other FPGA devices when the resource information of that FPGA is available. We anticipate p rogrammers will use this tool in the HLS development process. Note that our tool does not predict how to port a code to FPGA, but how much speedup is achievable if ported onto different FPGA devices.

Overview: Figure 2 illustrates the overview of the tool. XPPE is written in MATLAB and uses a neural network to estimate the speedup of an application for a target FPGA over ARM processor. XPPE gets HLS report as an input and extracts the information of the FPGA used for synthesize, and the resource utilization on that FPGA. Moreover, the user must provide the specification of a target FPGA. There is no limit on the specification of the target FPGA. Therefore, XPPE can be used for future FPGA devices and gives an insight to designers.

Artificial Neural Networks (ANNs) are a class of machine learning models that can map a set of input parameters to a set of target values. The model used in this tool is a 3-layer fully connected neural network with 900 hidden neurons. The inputs are available resources on target FPGA, resource utilization of application reported by HLS (extracted features as in Table 3), and characteristics of the application for cross-platform speedup estimation. The output is the speedup estimation for on the target FPGA over ARM A-9 processor.

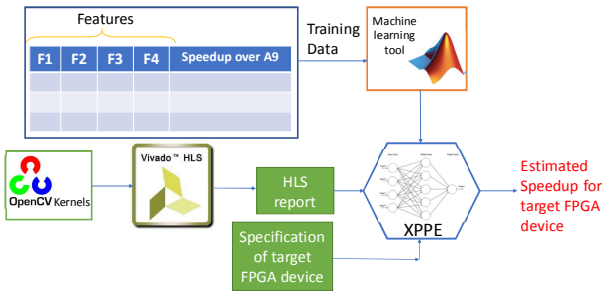

Figure 2: Overall flow of XPPE

Training and validation: MATLAB's Neural Net Fitting tool has been used for building the model. We trained the network with Levenberg-Marquardt algorithm. Figure 3 shows the correlation of output of network and the target value for different datasets namely, train, validate and the test data. The obtained correlation between the estimated and actual speedup for the test data is R=0.97. Figure 4 shows the histogram of error.

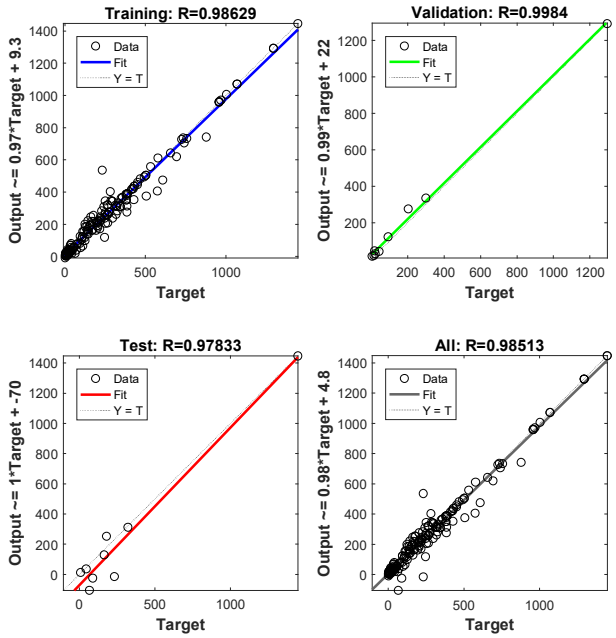

Figure 3: Neural network performance

To evaluate the accuracy of the estimator, we use equation (1) to calculate the percentage RMSE (Root Mean Squared Error) between the predictions and the real measurements.

$$RelativeRMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\frac{p_i - a_i}{a_i})^2} \times 100 \quad (1)$$

where N is the number of samples, and p_i and a_i are the predicted and actual values of the sample, respectively. We want the % relative RMSE to be as low as possible. RMSE is a standard metric in regression which is sensitive to scalability. For example, an RMSE of 1 s in runtime prediction is not acceptable if the actual runtime is 2 s, but can be acceptable if the actual runtime is 1000 s. Expressing the error as a percentage of the actual value solves this issue. The proposed neural network architecture has an mean square error of 5.1%. The model has an accuracy of 91%, 98%, and 95% for training, validation and testing respectively.

Assessment of Neural Network: Figure 4 shows how the number of hidden neurons changes the average accuracy of speed up estimation. Based on the results, we decided to fix the number hidden neurons to 900 for XPPE where the prediction accuracy is nearly 94.9% on Average.

4 DESIGN SPACE EXPLORATION

The goal of High-Level Synthesis design is to reduce the complexity of developing hardware accelerator and to increase the usage of reconfigurable devices in different domains. HLS helps a software developer to convert the high level code into hardware language without having deep knowledge of hardware. But, there exists a

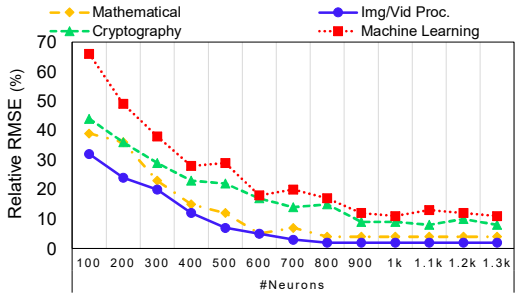

Figure 4: Error of neural network

decision making gap between a hardware design and its implementation on FPGA device. As various FPGA devices provide different capability and as a result different speedup. As software developers may don't have enough knowledge about choosing the best device for their implementation, it is a crucial step to first perform a design space exploration.

In this section, we show how XPPE can be used to explore the design space of hardware acceleration. To this goal, we first use XPPE to estimate the speedup of different applications on various types of FPGAs. In order to shed insight on the raw results, we analyze the importance of FPGA parameters on the acceleration speedup of applications.

In order to determine the importance of FPGAs' parameters, we employed linear regression on our date gathered from XPPE to extract the relation between different FPGA parameters and the speedup of applications. The coefficient of each FPGA parameter determines the impact of that parameter on the speedup (performance) of the accelerator. The FPGA parameters selected to build the relation for speedup are FPGA frequency, the number of logic cells, look-up-tables (LUTs), flip-flops (FF), Block RAMs (BRAMs), DSPs, and bandwidth of the interconnect between FPGA and DRAM. Figure 5 illustrates the visualization of coefficients. The observation reveals that different parameters of FPGA, depending on the application type, plays a crucial role in improving the accelerator's speed up.

To correctly interpret the results, we have to first understand the characteristics of each application. In our experiments, we observed that applications can be data-intensive, have multistage computation, or could be highly parallel. We consider an application as data-intensive if it several times transfers a large amount of data from DRAM to PL part through DMAs (direct memory access) or vice versa. Moreover, we refer to dependent kernels of an application that cannot work in parallel as multistage computation. To accelerate these part of an application, pipeline strategy or high level parallelism such as multiple instances of such kernels can be used. On the other hand, some kernels in an application can be implemented in a fully parallel manner and we can unroll those compute units to increase the throughput at the cost of LUTs/FFs. these kernels are called highly parallel kernels. We note that different HLS optimizations employed based on the target FPGA size.

The aforementioned characteristics directly impact on the resource requirements of an application. Therefore, the effectiveness

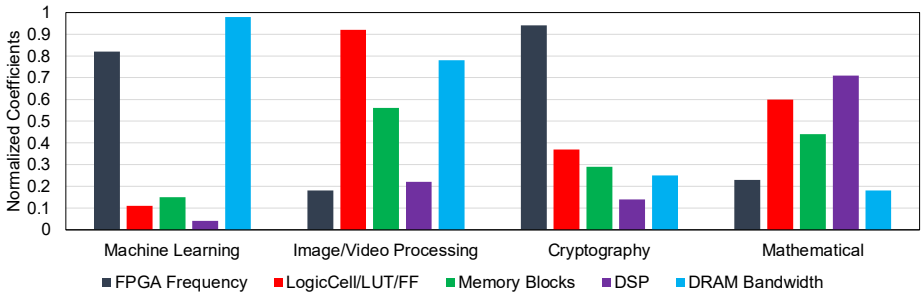

Figure 5: Parameter importance based on regression

of FPGAs' parameters on the performance improvement of acceleration depends on these characteristics in the applications. We identified these characteristics in each class of application and Table 4 illustrates them.

Table 4: Characteristics of applications

Application Type	Data Intensive	Multistage Computation	Highly Parallel
Machine Learning	✓✓✓	✓✓	
Image/Video Processing	✓✓		✓✓✓
Cryptography	✓	✓✓✓	✓
Mathematical		✓	✓✓

For machine learning applications, we observe that FPGA frequency and DRAM Bandwidth are the most important parameters. However, other parameters related to the number of resources is not crucial. DRAM Bandwidth is important for machine learning applications as they repeatedly read and write data from/to main memory. Therefore, the DRAM bandwidth becomes the bottleneck of the system. Moreover, the functionality of machine learning kernels implemented in the programmable logic is mostly sequential. Hence, there is no significant performance gain to use unrolling or multiple instances. As a result, the abundance of resources such as LUT or FF is not beneficial. In this case, one of the approaches to increase the speed up of the accelerator is to use the pipeline strategy that can improve the throughput of ML kernels. In the pipeline strategy, the period of the clock cycle is important for better throughput. Therefore, a faster FPGA with higher clock frequency outperforms other FPGAs in terms of performance in machine learning applications.

Unlike machine learning applications, resources such as LU/FF are the main contributors to the speed up of image and video processing applications. The functionality of these applications is mostly to apply a simple kernel or filter on the matrix of input data. As these tasks are independent, the process can be done concurrently on all part of input using multiple instances of such kernels and at the same time, it is possible to use the unrolling technique to process a larger chunk of data if the resource of FPGA allows. Moreover, these applications require to have data inside the programmable logic part that causes to use more memory blocks. Hence, this parameter is also important. Similar to the machine learning applications, image and video processing applications have a stream of data transfer to DRAM that causes contention in the memory interconnect. Therefore, we observe that DRAM bandwidth is again another important parameter in the performance of these type of accelerators.

The result shows FPGA frequency is the dominant parameter in determining the speed up of cryptography accelerators. These applications are mostly sequential and have multiple stages to compute the output results. The similar approach for machine learning applications can be applied here which is to use pipeline strategy and memory partitioning. We can use the same explanation here to justify why FPGA frequency is the most important parameter for the speed up these accelerators.

Eventually, for mathematical kernels, logic resources, memory blocks, and DSP units are the most influential parameters on the speed up. Because these kernels are mostly simple and parallel-able that extra resources can be used to improve the performance.

Discussion: Given the results and discussions provided in table 2 and Figure 5, respectively, following trends are observed with respect to application's characteristics and FPGA platform's configuration: Applications that have the potential of parallelism can benefit more from logical resources such as logic cells, LUTs, and FFs as directives (e.g. unrolling) can enhance the speed up. Therefore, we observed that a high-end FPGA improves the performance of image and video processing applications 17 times more than low-end FPGA. Moreover, applications that have more sophisticated computation kernels such as machine learning and cryptography benchmarks require a high frequency FPGA to increase the throughput. In this case, the amount of resources is not crucial and results show that a high-end FPGA only improves the performance 1.6× over medium-end FPGA on average. Applications that require to frequently access to data in DRAM have potential to easily saturate the memory bandwidth as the abundance of resources in FPGAs opens space for more computing units which increases the demand for data to feed in. Hence, for data intensive applications, it is important to balance the number of computing units as accelerators in PL part and DRAM bandwidth to prevent it becomes the bottleneck of the system.

5 RELATED WORK

The work in [24] proposed Aladdin, a pre-RTL, power-performance accelerator modeling framework and demonstrated its application to system-on-chip (SoC) simulation. Aladdin highlights impact of system-level parameters on accelerator design trade-offs when integrated with a standard cache and DRAM simulator and removes

synthesis and reuses optimization across a large design space for fast exploration among ASIC accelerators.

In [9], the performance prediction for Zynq-SoC is proposed which estimates the performance based on the execution time of an application on the FPGA. This is primarily used for partitioning the task onto hardware during hardware-software co-design. The authors in this work primarily presents a performance parallel heterogeneous estimation for systems for hardware/software co-design and run-time heterogeneous task scheduling.

The RC Amenability test (RAT) methodology proposed in [8] performs the performance prediction by exploiting the common algorithmic and architectural features of the application and FPGA. Although RAT supports the rapid exploration and prediction of strategic design trade-offs during the formulation stage of application development, it is confined to single-FPGA systems. In order to address this shortcoming, RATSS methodology [7] is proposed as an extension RAT that is applicable to multi-FPGA systems.

Vivado HLS will provide a very large range of cycles for variable loops, and the exact performance after each optimization becomes difficult to predict making the cycle analysis difficult. It is possible to estimate the performance of applications with variable bounds based on the software simulation flow as proposed in HLScope [2, 3]. The work in [3] proposes HLScope+ framework which is a high-level cycle estimation methodology for input-dependent FPGA designs using the HLS software simulation process.

The work in [28] analyzing the underlying architecture and common algorithmic features might not be feasible all the time. There are others work that looked into other aspects of hardware accelerators such as [14, 15]. Additionally, works in [12, 20–22, 25, 26] used machine learning to solve other issues in computer design. In contrast to the existing works, our proposed methodology (XPPE) predicts the performance of hardware accelerator that accounts for both the application characteristics and the FPGA platform parameters for prediction while it is not limited to any specific FPGA. In addition, the performance prediction of existing works [16] is based on a detailed analysis of application and architectural information of FPGA which reduces their usability, since extracting such information is time-consuming and also requires a deep knowledge of FPGA's architecture and hardware design.

6 CONCLUSIONS

We introduced XPPE, a neural network based tool for estimating the performance of an application on a different FPGA platform over ARM processor. Our evaluation results show that the correlation of estimated performance and actual speed up is more than 0.97. Moreover, we used XPPE to perform a design space exploration (DSE) of the FPGAs to determine the impact of FPGA parameters on the performance of diverse types of applications. It has been observed that applications that have the potential of parallelism can benefit more from logical resources on the high-end FPGAs. Moreover, applications that have more multistage computing kernels such as machine learning and cryptography benchmarks require a high frequency FPGA to increase the throughput. We observed that date-intensive kernels in applications can easily saturate the memory bandwidth. Hence, it is important to use medium or high-end FPGAs to prevent it becomes the bottleneck of the system for such applications.

7 ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 152691.

REFERENCES

[1] Kimia Zamiri Azar et al. 2019. SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019).
[2] Y. Choi and J. Cong. 2017. HLScope: High-Level Performance Debugging for FPGA Designs. In *FCCM*.
[3] Young-kyu Choi et al. 2017. HLScope+: Fast and accurate performance estimation for FPGA HLS. In *ICCAD*.
[4] Steve Dai et al. 2018. Fast and Accurate Estimation of Quality of Results in High-Level Synthesis with Machine Learning. In *FCCM*.
[5] Farnoud Farahmand et al. 2017. Minerva: Automated hardware optimization tool. In *ReConFig*.
[6] Yuko Hara et al. 2008. Chstone: A benchmark program suite for practical c-based high-level synthesis. In *ISCAS*.
[7] Brian Holland and et.al. 2011. An Analytical Model for Multilevel Performance Prediction of Multi-FPGA Systems. *ACM Trans. Reconfigurable Technol. Syst.* 4, 3 (Aug 2011), 27:1–27:28.
[8] Brian Holland, Karthik Nagarajan, and Alan D. George. 2009. RAT: RC Amenability Test for Rapid Performance Prediction. *ACM Trans. Reconfigurable Technol. Syst.* 1, 4 (Jan 2009), 22:1–22:31.
[9] Daniel Jiménez-González and et.al. 2015. Coarse-Grain Performance Estimator for Heterogeneous Parallel Computing Architectures like Zynq All-Programmable SoC. *InternationalWorkshop on FPGAs for Software Programmers* (2015).
[10] David Koeplinger et al. 2016. Automatic generation of efficient accelerators for reconfigurable hardware. In *ISCA*.
[11] Hosein Mohammadi Makrani et al. 2018. Compressive Sensing on Storage Data: An Effective Solution to Alleviate I/0 Bottleneck in Data-Intensive Workloads. In *IEEE ASAP*.
[12] Hosein Mohammadi Makrani et al. 2018. Energy-aware and Machine Learning-based Resource Provisioning of In-Memory Analytics on Cloud. In *SoCC*.
[13] Hosein Mohammadi Makrani and Houman Homayoun. 2017. MeNa: A memory navigator for modern hardware in a scale-out environment. In *IISWC*.
[14] Katayoun Neshatpour et al. 2018. Architectural considerations for FPGA acceleration of Machine Learning Applications in MapReduce. In *SAMOS*.
[15] Katayoun Neshatpour et al. 2018. Design Space Exploration for Hardware Acceleration of Machine Learning Applications in MapReduce. In *FCCM*.
[16] Craig P Steffen. 2007. Parametrization of algorithms and FPGA accelerators to predict performance. *Proc. Reconfigurable System Summer Institute* (2007).
[17] Brandon Reagen et al. 2014. Machsuite: Benchmarks for accelerator design and customized architectures. In *IISWC*.
[18] S. Rezaei and et.al. 2016. Data-rate-aware FPGA-based acceleration framework for streaming applications. In *ReConFig*.
[19] Shervin Roshanisefat et al. 2018. SRCLock: SAT-Resistant Cyclic Logic Locking for Protecting the Hardware. In *GLSVLSI*.
[20] Hossein Sayadi et al. 2017. Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures. In *ICCD*.
[21] Hossein Sayadi et al. 2018. Customized machine learning-based hardware-assisted malware detection in embedded devices. In *TrustCom/BigDataSE*.
[22] Hossein Sayadi et al. 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *DAC*.
[23] Benjamin Carrion Schafer et al. 2014. S2CBench: Synthesizable SystemC benchmark suite for high-level synthesis. *IEEE Embedded Systems Letters* (2014).
[24] Y. S. Shao et al. 2014. Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures. In *ISCA*.
[25] J. Stangl et al. 2018. A Fast and Resource Efficient FPGA Implementation of Secret Sharing for Storage Applications. In *DATE*.
[26] Ashkan Vakil et al. 2019. IR-ATA: IR Annotated Timing Analysis, A Flow for Closing the Loop Between PDN design, IR Analysis Timing Closure. In *ASP-DAC*.
[27] Guanwen Zhong et al. 2017. Design Space exploration of FPGA-based accelerators with multi-level parallelism. In *DATE*.
[28] G. Zhong and et al. 2016. Lin-Analyzer: A high-level performance analysis tool for FPGA-based accelerators. In *DAC*.
[29] Yuan Zhou et al. 2018. Rosetta: A Realistic High-Level Synthesis Benchmark Suite for Software Programmable FPGAs. In *FPGA*.