Accurate Performance and Power Prediction for FPGAs Using Machine Learning

Lina Sawalha Tawfiq Abuaita Martin Cowley Sergei Akhmatdinov Adam Dubs
Western Michigan University, Kalamazoo 49009
Email: {lina.sawalha, tawfiq.s.abuaita, martin.j.cowley, sergei.akhmatdinov, adam.p.dubs} @wmich.edu

Introduction: Although high-level synthesis (HLS) tools have allowed software engineers to investigate FPGAs, they are slow to synthesize and simulate, and they require users' knowledge and setup time. Using machine learning algorithms (ML) to predict applications' execution time and power consumption on FPGAs can significantly speed up the process. Oneal et al. [1] used random forest along with CPU code and its microarchitecture-dependent runtime features to predict the performance and power of FPGAs. However, they split benchmarks into several windows of execution time (data points), which can be similar. Many similar data points in the dataset result in a model that may not generalize well for new applications. In this work, we propose a fast, accurate, and generalizable method to predict the execution time, and power consumption of applications on FPGAs using ensemble ML. Our method uses CPU code and related features at three levels LLVM-IR, source code, and dynamic runtime. We use cross-validation and ensure that our method is accurate, robust, and generalizable.

Methodology: We used Legup 4.0 HLS tool for synthesizing C/C++ benchmarks. We modified 19 benchmarks to work with Legup: all 12 CHStone [2], three SHOC [3] and four Rodinia [4] benchmarks. To collect the target metrics of execution time and power consumption for each benchmark, we performed post place and route synthesis using Quartus II, on the Verilog code generated by Legup, as shown in Figure 1. We use CPU code and related features at the source code, microarchitecture-independent [6], and instruction-set-architecture independent (using LLVM IR [5]) levels, a total of 132 features, along with ML algorithms to predict the power consumption and execution time on FPGAs. The features are combined in one dataset that is cleaned and then fed to our ML-based feature selection model. Feature selection chooses features at different levels to achieve higher accuracy. The selected features are fed to the metric prediction model as input features along with the corresponding target performance/power metrics. We use Python 3.6.9, and Scikit-learn 0.24 library.

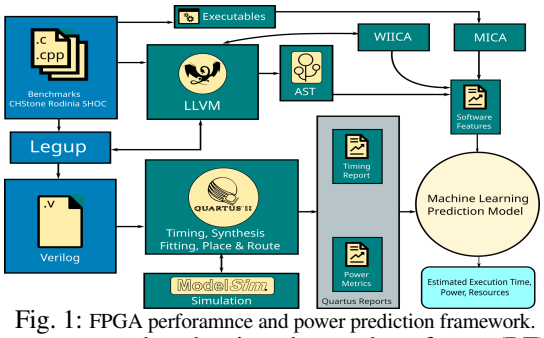

Fig. 1: FPGA perforamnce and power prediction framework.

Features are reduced using the random forest (RF) algorithm during the feature selection process. Then we used the $SelectKBest$ algorithm to rank the features and select the top 18 ranked features. After that, an exhaustive search is used to find the best features for each ML algorithm. Because of the small dataset, to avoid overfitting we clustered the dataset into three clusters based on the register-age, and store instructions features as two of the highest-ranked features. We compared different ML algorithms including linear and exponential regression (LR & ER), RF, decision tree (DT), K-nearest neighbor (KNN), and artificial neural network (ANN). We validated our results using 5-fold cross-validation, ensuring no overfitting for both feature selection and prediction.

Results: Figure 2 shows the out-of-sample percent error of the number of clock cycles (CC), clock period (CP), execution time (ET), and power consumption (P) using exponential regression as the most accurate algorithm. The results show a prediction with percentage errors as low as 0.35%, 0.1%, 2%, and 0.2% for CC, CP, ET, and P respectively. The average errors are 2.2%, 10.3%, 11.14%, and 2.7% respectively. Table I compares the average error rate of 100 runs for different ML algorithms. It also reports HLSPredict [1] results for our dataset using all $Likwid$ performance-monitoring-counter features and the RF algorithm.

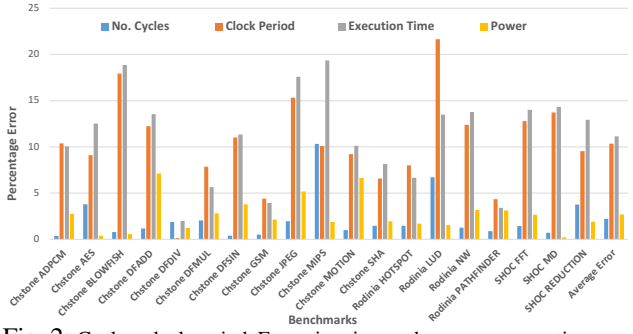

Fig. 2: Cycles, clock period, Execution time and power consumption error.

TABLE I: Average percentage errors of target features.

Algorithm	Clock Period	Total Power	Clock Cycles
ER	10.35	2.67	2.20
LR	10.1	4.72	9.97
RF	20.47	8.47	203.15
ANN	107.42	58.6	48.42
KNN	17.97	8.68	517.34
DT	20.17	9.94	117.31
HLSP [1]	27.47	10.08	127.14

Acknowledgement: This paper is based on work funded by NSF award no. 1821691.

REFERENCES

[1] K. O'Neal et al., "HLSPredict: Cross Platform Performance Prediction for FPGA High-Level Synthesis," in *ICCAD*, 2018, pp. 1–8.
[2] Y. Hara, et al., "Chstone: A benchmark program suite for practical c-based high-level synthesis," in *ISCS*, 2008, pp.1192–1195.
[3] V. Tipparaju, and J. S. Vetter, A. Danalis et al., "The scalable heterogeneous computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on GPCGPU*, 2010, pp. 63–74.
[4] K. Skadron, S. Che et al., "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2009, pp. 44–54.
[5] Y. S. Shao and D. Brooks, "ISA-independent workload characterization and its implications for specialized architectures," in *ISPASS*, 2013, pp. 245–255.
[6] K. Hoste and L. Eeckhout, "Microarchitecture-independent workload characterization," *IEEE Micro*, vol. 27, no. 3, pp. 63–72, 2007.

978-1-6654-8332-2/22/$31.00