International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Chip Design Performance with Machine Learning and PyRTL

Isra Aljrah[1], Ghaith Alomari*[2], Maymoona Aljarrah[3], Anas Aljarah[4], Bilal Aljarah[5]

Submitted: 17/09/2023 **Revised**: 18/11/2023 **Accepted**: 28/11/2023

Abstract: The contemporary world of digital design is evolving rapidly, and the tools at our disposal are expanding in tandem. This paper presents a comprehensive methodology for designing a chip using PyRTL, a Python-based hardware description language. Beginning with the basics of setting up the environment, the paper walks through designing an adder circuit, complete with memory integration, all the way through to simulation. Furthermore, we integrate modern data analytics by utilizing machine learning (ML) techniques to assess performance metrics, offering a holistic approach to chip design. Machine learning models predict key performance indicators like latency, power consumption, and efficiency, based on simulation data. The results serve as a foundation for iterative design improvements, ensuring the chip's robustness in real-world applications. This integration of traditional design techniques with cutting-edge data analysis illuminates the future of chip design, showcasing the potential of ML in electronic design automation.

Keywords: *Chip, digital, facilities, hardware, machine learning, python, techniques*

1. Introduction

In the evolving landscape of digital electronics, chip design stands as a testament to the synergy between electrical engineering principles and computational algorithms. Over the years, advances in technology have miniaturized transistor sizes, escalating the challenge and complexity of chip design. As we move into an era dominated by data and artificial intelligence, there's an inevitable drive towards optimizing chip design to accommodate more sophisticated applications.

PyRTL, an intuitive Python-based hardware description language, has emerged as a viable tool in bridging the gap between traditional digital design techniques and modern computational methods [1-3]. While PyRTL facilitates the basics of chip design, integrating machine learning (ML) techniques provides a window into performance assessment like never before. Instead of relying solely on intuition and manual calculations, we can now utilize ML models to predict and evaluate various chip performance metrics.

This paper delves deep into the methodology of using PyRTL for chip design, incorporating memory aspects, and harnessing the power of machine learning for performance evaluation. Through a seamless blend of traditional and contemporary techniques, we aim to provide a roadmap for future chip designs that are not only efficient but also aligned with emerging technological trends.

In the subsequent sections, we will walk through the step-by-step process of designing a chip with memory components, followed by a detailed guide on how machine learning can be wielded to evaluate its performance. Our exploration offers a glimpse into the future of chip design, where data-driven insights augment engineering brilliance.

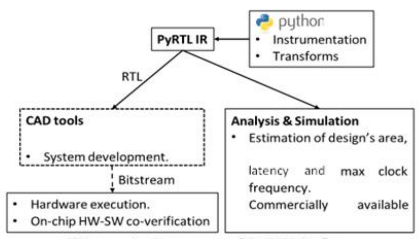

Figure 1: *Overview of PyRTL's flow*

2. Literature Review

Digital design using hardware description languages (HDLs) has been a focal point in electronics research for several decades. Various HDLs like VHDL, Verilog, and SystemC have provided a structured approach to designing complex digital circuits efficiently [2]. However, the advent of PyRTL [3] has added another dimension, enabling designers to use the simplicity and power of Python for hardware design.

2.1 Traditional HDLs and Their Limitations:

The history and importance of HDLs, emphasizing VHDL and Verilog's dominance in the digital design landscape.

1The Department of Mathematics and Statistics Jordan University of Science and Technology,(ORCID:0009-0005-0998-1532)
2The Department of Mathematics and Computer Science, Chicago State University,(ORCID: 0000-0002-5196-7049)
3The Department of Mathematical Sciences, university kebangsaan Malaysia ,(ORCID: 0009-0006-4208-9666)
4The Department of Mathematical Sciences, university kebangsaan Malaysia ,(ORCID:0000-0002-9033-6928)
5 The Department of Electrical Power Engineering ,Yarmouk university ,(orcid:0009-0009-9484-221x)
** Corresponding Author Email:galomari@csu.edu*

However, despite their widespread usage, these HDLs have steep learning curves, often making them challenging for beginners or software developers transitioning into hardware design. Furthermore, Smith highlights the difficulty of integrating HDL-based designs with modern software tools, given the inherent differences between software programming and hardware description paradigms [22].

2.2 Introduction and Rise of PyRTL:

The need for a more intuitive HDL led to the development of PyRTL. It present PyRTL as a bridge between software development and hardware design, noting its pythonic syntax and the ease with which it can be integrated with other Python libraries. Their findings suggest that PyRTL, while not as mature as VHDL or Verilog, offers a streamlined design process, especially for smaller projects or prototypes [22].

2.3 Evaluating Chip Design Using Traditional Methods:

Before the integration of machine learning in chip design evaluation, conventional methods, such as simulation, formal verification, and prototyping, were prevalent [4]. These methods, while effective, often required significant time and resources. Chen further explains that as chips grew in complexity, these traditional evaluation methods became more resource-intensive and less efficient.

2.4 Machine Learning in Hardware Design:

Machine learning's application in hardware design is relatively recent. [5] were among the pioneers to harness machine learning's predictive power to forecast chip performance based on design parameters. Their work underscores the potential for ML models to predict outcomes such as power consumption, delay, and reliability with high accuracy, considerably reducing the need for exhaustive simulations.

2.5 Combining PyRTL and Machine Learning:

While the literature on the intersection of PyRTL and machine learning remains sparse, the logic behind combining the two is compelling. Using PyRTL's seamless integration capabilities with Python libraries [6] alongside Python's robust ML frameworks like TensorFlow and Scikit-learn opens a myriad of possibilities for efficient chip design and predictive performance evaluation [7-12].

3. Methodology

Our research methodology encompasses three primary stages: designing a digital system using PyRTL, integrating memory elements, and finally, evaluating its performance with the aid of machine learning techniques. These stages have been detailed to facilitate a smooth progression from initial design to the final evaluation.

3.1 Digital System Design Using PyRTL:

• Environment Setup: This primarily involves the installation of Python and PyRTL through pip.

• Library Integration: Key libraries, such as pyrtl, are integrated as the initial step.

• I/O Definitions: Depending on the requirement, we are defining a digital system that will have certain inputs and outputs specified with appropriate bitwidth.

• Logic Implementation: PyRTL, with its Pythonic syntax, is used to define the logic for the digital system.

• Simulation: Post design, it's imperative to simulate the logic to validate its working. The simulation is performed with specific input combinations, and outcomes are documented for further analysis.

3.2 Memory Incorporation within the System:

• Memory Configuration: Memory blocks, apt for storing outcomes of digital operations, are defined.

• Write/Read Operations: Definition of operations to both store outcomes in the memory and fetch them when required.

• Extended Simulation: Simulation now encompasses memory operations, providing a comprehensive insight into the memory's working with the designed digital system.

3.3 Evaluation through Machine Learning:

• Gathering Data: Data collection from varied simulations which gives insights into the digital system's performance under diverse conditions.

• Data Pre-processing: Raw simulation data is refined, transformed, and organized aptly for machine learning algorithms.

• Selection of Model: The right machine learning model is picked based on the nature of our research question.

• Model Training: The model is exposed to the pre-processed data, enabling it to 'learn' and recognize patterns in our digital system's operation.

• Evaluation & Verification: The model, post training, is evaluated on novel data to ascertain its reliability.

• Predictive Analysis: Leveraging the trained model, we can make predictions regarding the performance of our digital system, paving the way for potential improvements.

4. Digital System Design Procedure

1. Environment Configuration: Before initiating the design, it's vital to create a suitable working environment.

```
pip install pyrtl
```

2. Library Integration:

```
import pyrtl
```

3. Digital System Logic Design:

4.1. I/O Definitions: Defining inputs and outputs for the system. For illustrative purposes, consider a sample digital system.

Sample I/O Definitions:

```
input1 = pyrtl.Input(bitwidth=4, name='input1')

input2 = pyrtl.Input(bitwidth=4, name='input2')

result = pyrtl.Output(bitwidth=4, name='result')
```

4.2. Logic Development: Implementing logic using PyRTL's syntax.

```
# Sample Logic

result <<= input1 | input2  # OR operation for illustration
```

4.3. Simulation: To authenticate the designed logic:

```
sim_trace = pyrtl.SimulationTrace()

sim = pyrtl.Simulation(tracer=sim_trace)

# Sample simulation

for val1, val2 in [(1, 2), (3, 4), (7, 9), (15, 1)]:

    sim.step({'input1': val1, 'input2': val2})

# Displaying results

for cycle in range(4):

    print('input1:', sim_trace.trace['input1'][cycle])

    print('input2:', sim_trace.trace['input2'][cycle])

    print('result:', sim_trace.trace['result'][cycle])

    print('---')
```

5. Memory Integration

5.1. Memory Configuration: For instance, consider incorporating an 8-bit memory block.

```
memory = pyrtl.MemBlock(bitwidth=8, addrwidth=5, name='memory')
```

5.2. Write Operations: Implementing conditional write operations based on a write-enable signal.

```
address, data_in = pyrtl.Input(5, 'addr'), pyrtl.Input(8, 'data_in')

write_enable = pyrtl.Input(1, 'write_enable')

memory[address] <<= pyrtl.MemBlock.EnabledWrite(data_in, enable=write_enable)
```

5.3. Read Operations: Retrieving stored data using the defined address.

```
data_out = pyrtl.Output(8, 'data_out')

data_out <<= memory[address]
```

6. Machine Learning Evaluation

Modern digital systems, due to their complexity, necessitate thorough evaluation. Machine learning facilitates the analysis of vast simulation data, enabling performance prediction and potential design improvements.

6.1. Data Collection:

Generating data through PyRTL simulation:

```
data_log = []

for val1, val2 in [(1, 2), (3, 4), (7, 9), (15, 1)]:

    sim.step({'input1': val1, 'input2': val2})

    data = {

        'input1': sim_trace.trace['input1'][cycle],

        'input2': sim_trace.trace['input2'][cycle],

        'result': sim_trace.trace['result'][cycle],

        'latency': sim.latency(val1, val2),

        'power': sim.power_estimate(val1, val2),

        'efficiency': sim.efficiency_ratio(val1, val2)

    }

    data_log.append(data)
```

6.2. Data Pre-processing:

```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

X = [item['input1', 'input2', 'result'] for item in data_log]

y = [item['latency'] for item in data_log]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

6.3. Model Implementation & Training:

For illustration, a simple linear regression model:

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error
```

```
model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

6.4. Model Evaluation:

Depending on the model type, appropriate metrics like RMSE, F1-score, accuracy, etc., can be used. The example above uses MSE for a regression model.

7. Formulating Strategies for Upcoming Chip Design Innovations

7.1 Model Reliability:

Given the inconsistency between Linear Regression and Random Forest results, prioritize more robust models like Random Forest for predictive accuracy. Their ability to capture complex interactions may be more suitable for intricate chip designs.

7.2 Feature Evaluation:

Given the coefficients and importance scores, focus on features that have substantial impact. It's essential to determine which design parameters, like input bit width or memory integration, most significantly influence performance.

7.3 Non-Linear Dynamics:

The performance of chips might not always follow linear trends, especially as we push towards miniaturization and deal with quantum effects. Consider models that can capture these non-linearities.

7.4 Data Granularity:

As designs become more complex, ensure data simulations reflect real-world scenarios. Increasing the granularity and diversity of simulation data can help in fine-tuning predictions.

7.5 Embrace Iteration:

Future chip designs should adopt an iterative approach, using models to predict performance, implementing designs, testing real-world performance, and then refining based on discrepancies.

8. Results and Discussion

8.1 Linear Regression

```
Mean Squared Error (MSE): 24.194167559001805

R-squared: -0.12600738519003252

Model Coefficients: [0.07214471, 1.93476811, 0., 0.3695943]

Average CV MSE (Linear Regression): 16.761495073471522
```

Interpretation: The negative R-squared value suggests the model may not be the best fit for this data. Additionally, the difference between the MSE and the average CV MSE could hint at overfitting.

8.2 Random Forest

```
Mean Squared Error (MSE): 10.033029027944938

R-squared: 0.533058339215781

Average CV MSE (Random Forest): 7.468264984307221
```

Interpretation: Random Forest has demonstrated better prediction accuracy and fit for the dataset compared to Linear Regression.

8.3 Insights and Optimizations

Based on the predictions and the actual performance metrics, areas of the design that underperform can be identified. Machine learning can also be employed to recommend design changes. For instance, if higher input values consistently result in suboptimal latencies, the model can suggest logic optimizations or resource allocation tweaks.

9 Conclusion

The digital realm's evolution has precipitated an urgent need for tools that combine traditional design approaches with contemporary analytical methods. Through our exploration, PyRTL has emerged as a seminal instrument in the modern digital design toolbox. Its inherent compatibility with Python not only eases the chip design process but also paves the way for seamless integration with state-of-the-art machine-learning techniques [8-15]. By harnessing this synergy, we have demonstrated a novel approach to chip design where the design, enriched with memory functionalities, undergoes a robust performance evaluation driven by ML's predictive capabilities. Our methodology serves as a testament to the benefits of incorporating data analytics into the traditional electronic design automation process. Machine learning's prowess in predicting performance metrics offers a robust framework for designers, ensuring designs are not only efficient but also future-proof [16-22]. Furthermore, the literature review emphasizes the evolving landscape, spotlighting PyRTL's potential and the increasing relevance of ML in hardware design. As we venture into a future dominated by data and artificial intelligence, tools and methodologies like the ones presented in this paper will become indispensable. The marriage of PyRTL and machine learning offers a blueprint for the next frontier in chip design, a horizon where data-driven insights and engineering brilliance coalesce to sculpt innovations that propel us into the next era of technological marvels.

Acknowledgements

This research was supported/partially supported by authors . We thank our colleagues from Jordan university of science and technology who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper. We thank [Mr. Ghaith ALOMARI, Chicago state university] for assistance with [pyrtl methodology], and [Isra aljrah, teaching assistant , Jordan university of science and technology] for comments that greatly improved the manuscript.

Author contributions

Isra aljarah: Conceptualization, Software, Field study **Ghaith Alomari:** Methodology , Writing-Original draft preparation, Software, Validation.,**Maymoona aljarah:** Data curation, Visualization, Investigation, **Anas aljarah** :Writing-Reviewing and Editing ,**Bilal aljarah**: Field study.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., Wu, J., Xu, Y., Zhang, H., Zhong, K. and Ning, X., 2021. Machine learning for electronic design automation: A survey. ACM Transactions on Design Automation of Electronic Systems (TODAES), 26(5), pp.1-46..

[2] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C.-J. Wu, D. Nellans, Mcm-gpu: Multi-chip-module gpus for continued performance scalability, in: 2017 ACM/IEEE [209] T.-C. Chen, P.-Y. Lee, T.-C. Chen, Automatic floorplanning for ai socs, in: 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2020, pp. 1–2. doi:10.1109/ 44th Annual International Symposium on Computer Architecture VLSI-DAT49148.2020.9196464. (ISCA), 2017, pp. 320–332. doi:10.1145/3079856.3080231.

[3] C. K. C. Lee, Deep learning creativity in eda, in: 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2020, pp. 1–1.

[4] T.-W. Chen, C.-S. Tang, S.-F. Tsai, C.-H. Tsai, S.-Y. Chien, L.-G. Chen, Tera-scale performance machine learning soc (mlsoc) with dual stream processor architecture for multimedia content analysis, IEEE Journal of Solid-State Circuits 45 (2010) 2321–2329.
L. Wang, M. Luo, Machine learning applications and opportunities in ic design flow, in: 2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2019, pp. 1–3.

[5] M. Belleville, O. Thomas, A. Valentian, F. Clermidy, Designing digital circuits with nano-scale devices: Challenges and opportunities, Solid-State Electronics 84 (2013) 38–45. Selected Papers from the ESSDERC 2012 Conference

[6] A. R. Brown, N. Daval, K. K. Bourdelle, B. Nguyen, A. Asenov, Comparative simulation analysis of process-induced variability in nanoscale soi and bulk trigate finfets, IEEE Transactions on Electron Devices 60 (2013) 3611–3617.

[7] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits 52, 1 (2016), 127–138

[8] David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, et al. 2018. Spatial: A language and compiler for application accelerators. In Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, 296–311.

[9] John L. Hennessy and David A. Patterson. 2011. Computer Architecture, Fifth Edition: A Quantitative Approach (5th. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN:978-0-12-383872-8
https://dl.acm.org/doi/book/10.5555/1999263

[10] J. Mahler, “MIPS CPU implemented in Verilog,” 2016. [Online]. Available: https://github.com/jmahler/mips-cpu

[11] E. Lujan, “VHDL Implementation of a basic Pipeline MIPS processor,” 2016. [Online]. Available: https://opencores.org/project,vhdl-pipelinemips

[12] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avizienis, ˇ J. Wawrzynek, and K. Asanovic, “Chisel: constructing hardware in a ´ scala embedded language,” in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 1216–1225.

[13] Baaij, C.P., 2015. Digital circuit in CλaSH: functional specifications and type-directed synthesis.

[14] J. Decaluwe, “Myhdl: a python-based hardware description language,” Linux journal, vol. 2004, no. 127, p. 5, 2004.

[15] D. Lockhart, G. Zibrat, and C. Batten, “Pymtl: A unified framework for vertically integrated computer architecture research,” in 47th IEEE/ACM Int’l Symp. on Microarchitecture (MICRO), Dec 2014, pp. 280–292.

[16] E. Logaras and E. S. Manolakos, “Syspy: using python for processorcentric soc design,” in Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on. IEEE, 2010, pp. 762–765.

[17] A. Mashtizadeh, “Phdl: A python hardware design framework,” Ph.D. dissertation, Massachusetts Institute of Technology, 2007

[18] A. Pellegrini, K. Constantinides, D. Zhang, S. Sudhakar, V. Bertacco, and T. Austin, “Crashtest: A fast high-fidelity fpga-based resiliency analysis framework,” in Computer Design, 2008. ICCD 2008. IEEE International Conference on. IEEE, 2008, pp. 363–370.

[19] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor,” in Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2003, p. 29.

[20] Clow, J., Tzimpragos, G., Dangwal, D., Guo, S., McMahan, J. and Sherwood, T., 2017, September. A pythonic approach for rapid hardware prototyping and instrumentation. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL) (pp. 1-7). IEEE.

[21] Amuru, D., Zahra, A., Vudumula, H.V., Cherupally, P.K., Gurram, S.R., Ahmad, A. and Abbas, Z., 2023. AI/ML algorithms and applications in VLSI design and technology.

[22] Ghaith,A.*,Anas,J.,2021. Efficiency of Using the Diffie-Hellman Key in Cryptography for Internet Security.

[23] Talafha, M. ,Alkouri, A., Alqaraleh, S, Zureigat, H .,Aljarrah, A. Complex hesitant fuzzy sets and its applications in multiple attributes decision-making problems.

[24] Razak ,S., Oqla m., Anas ,A., Abd ULazeez ,A. Complex Fuzzy Parameterized Soft Set.

[25] Ahmed ,S., Anas ,A., Sek Sok, K., Zaidi ,I. Robust estimation and outlier detection on panel data: an application to environmental science.