Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Exploring Machine Learning Algorithms to Boost Functional Verification: A Comprehensive Survey

Dharmveer Singh Rajpoot[1], Manasa Adusumilli[2], Priyanka S Talekar[3], Muhammad Shameem P[4], Dr. D. Usha Rani[5], Sadik Khan[6]

[1]Professor, Computer Science Engineering and Information Technology, Jaypee Institute of Information Technology, Noida, India.
[2]Assistant professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India.
[3]Assistant Professor, Artificial Intelligence and Machine Learning, CMR Institute of Technology, Bangalore, India.
[4]Assistant Professor, Computer Science & Engineering, Madanapalle Institute of Technology & Science, Kadiri Road Angallu, Village, Madanapalle, Andhra Pradesh, India.
[5]Associate Professor, Department of Computer Science and Applications Koneru Lakshmaiah Education Foundation Vaddeswaram Andhra Pradesh, India.
[6]Assistant Professor, Department of Computer Science & Engineering, Institute of Engineering & Technology, Bundelkhand University, Jhansi, India.
E-mail: drdharmveer16382@gmail.com

Functional verification is essential in hardware design, ensuring systems meet their specifications before fabrication. As hardware complexity grows, traditional methods become increasingly resource-intensive, often leading to incomplete coverage and missed bugs. Machine learning (ML) presents a solution by automating test case generation, predicting outcomes, and improving coverage closure. This research explores the effectiveness of ML algorithms in enhancing functional verification, drawing on data from 120 respondents. Using statistical methods like ANOVA and Chi-square tests, the study demonstrates that ML-based verification significantly reduces verification time, increases coverage closure rates, improves bug detection, and lowers overall costs. The findings suggest that integrating ML into verification workflows can streamline the process, making it faster and more accurate while reducing resource expenditure.

Keywords: Functional verification, machine learning, hardware design, test case generation, coverage closure, bug detection, verification efficiency, ANOVA, Chi-square tests, cost reduction

1. Introduction

Functional verification is a critical step in the hardware design cycle, ensuring that the

designed system operates as intended and aligns with its specifications before it undergoes fabrication. The importance of this process cannot be overstated because undetected design flaws can result in significant financial losses and delays, especially when hardware errors are discovered late in the development cycle or after deployment. Functional verification is not only about finding flaws but also about confirming that the design meets all functional requirements in diverse operating conditions. This involves testing how the system handles various input scenarios and ensuring that every logic gate in a complex circuit performs its task as expected (Fine et al., 2006).

As modern hardware systems grow in complexity, with designs now encompassing millions or even billions of logic gates and increasingly intricate functionalities, traditional verification methods are becoming inadequate. These traditional techniques, including manual verification and constrained random verification, are extremely resource-intensive. They require substantial computational power, time, and human oversight, making it increasingly difficult for verification teams to meet project deadlines. Often, these methods fail to fully explore all possible design states, which leads to incomplete coverage and a higher risk of undetected bugs (Blanchette et al., 2016). This limitation is exacerbated by the increasing demand for faster time-to-market, as hardware designs need to evolve quickly to keep up with technological advancements and market competition.

The increasing pressure on verification teams has led to the exploration of new technologies that can automate and accelerate the verification process. Machine learning (ML) has emerged as a promising solution to address these challenges. ML algorithms, with their ability to learn from data and adapt their predictions and actions over time, can optimize many aspects of the verification process. They offer the potential to automate the generation of test cases, predict verification outcomes, and identify areas of the design that are more likely to contain errors, thus reducing both the time and resources required for thorough verification (Abd El Ghany & Ismail, 2021).

The aim of this study is to investigate how machine learning can enhance functional verification in modern hardware designs. By reviewing existing literature and conducting an empirical analysis involving 120 respondents from the hardware verification industry, this research seeks to evaluate the effectiveness of ML in improving verification outcomes. Specifically, we examine whether machine learning can increase coverage closure—ensuring that all potential functional states of the hardware design are tested—reduce the time required for verification, and enhance overall verification accuracy. We employ various statistical methods, including ANOVA, Chi-square tests, and T-tests, to analyze the data and assess the relationship between ML-based verification methods and traditional approaches.

2. Literature Review

2.1 Functional Verification Challenges

Functional verification is one of the most time-consuming stages of hardware development, often accounting for as much as 70% of the total project timeline (Fine et al., 2006). This figure reflects the enormous complexity of modern hardware systems and the need for exhaustive testing to ensure that no errors remain in the final product. The goal of functional verification

is to confirm that the design operates as specified under all possible conditions and inputs, a task that becomes more difficult as the design's complexity increases.

One of the primary challenges in functional verification is managing the complexity of modern designs. As hardware systems evolve to include millions of gates, intricate logic functions, and multiple layers of integration, the number of potential states that need to be tested grows exponentially. Each additional gate or logic component introduces new variables, increasing the difficulty of generating test cases that adequately cover all functional states. In particular, designs that incorporate new technologies like artificial intelligence or edge computing capabilities demand even more rigorous verification due to their novel and complex behaviors (Gad et al., 2021).

The second major challenge is long simulation times. Traditional verification techniques, such as constrained random verification, require substantial computational power and time to generate and simulate test cases. This method randomly selects input values to stimulate the hardware design, hoping to explore a wide range of functional states. However, random testing can be inefficient because many of the test cases generated do not provide meaningful new coverage, and the random nature of the inputs means that important edge cases might be missed altogether. As designs become more complex, the number of required simulations grows, and the process can take days, weeks, or even months to complete (Abd El Ghany & Ismail, 2021).

Another critical challenge is incomplete coverage, often referred to as coverage gaps. Even after long simulation periods, traditional methods often fail to achieve full coverage, meaning that some parts of the design remain untested. Coverage gaps occur when certain functional states or input conditions are not exercised during verification, leaving the design vulnerable to bugs that could surface in later stages of development or in field deployments. These bugs can be costly to fix if discovered after the product has been manufactured and deployed, and they can lead to hardware recalls, legal liabilities, and reputational damage (El Mandouh et al., 2018).

Given these challenges, the need for innovative solutions that can streamline the verification process while ensuring comprehensive coverage is clear. Machine learning offers a potential path forward, providing tools to optimize test generation, reduce simulation times, and improve overall coverage by learning from past verification efforts and identifying patterns that humans or traditional algorithms might miss.

2.2 Machine Learning in Functional Verification

Machine learning has gained attention as a powerful tool for addressing the significant challenges associated with functional verification. By leveraging historical data and learning from previous verification results, ML algorithms can optimize many aspects of the verification process, including predicting verification outcomes, optimizing test sequence generation, and accelerating coverage closure. This section reviews recent studies that highlight the benefits of ML in functional verification and explores how these models are being applied to solve some of the most pressing issues in this field.

One of the key areas where ML has demonstrated potential is in predicting verification outcomes. Instead of relying solely on random or exhaustive test case generation, ML

algorithms can be trained to predict which areas of the design are more likely to contain bugs. For example, supervised learning models can analyze past verification results and identify patterns associated with bugs or coverage gaps. These models can then be used to prioritize test cases that are more likely to reveal issues, thereby reducing the time spent on unproductive tests and increasing the likelihood of detecting critical bugs early in the verification process (Blanchette et al., 2016). By using historical verification data, ML models can also help engineers focus their efforts on parts of the design that have proven problematic in previous projects, streamlining the verification process.

Another promising application of ML is in test sequence generation. Traditional constrained random verification generates input stimuli randomly, which can lead to inefficient testing and coverage gaps. In contrast, ML techniques such as clustering and regression models can identify patterns in successful test cases and use that information to generate more targeted and efficient test sequences (El Mandouh et al., 2018). For instance, clustering techniques can group similar functional states or design behaviors together, allowing ML models to prioritize generating test cases that explore unique or unexplored states, rather than redundantly testing similar ones. This can significantly reduce the number of test cases required to achieve high coverage and improve the efficiency of the verification process.

ML has also been applied to accelerating coverage closure, a key goal in functional verification. Coverage closure refers to the process of ensuring that all possible functional states of a design have been exercised and tested. Given the complexity of modern hardware, achieving full coverage is a daunting task. However, ML models can be trained to detect areas of the design that have not been adequately tested and generate specific test cases to close those coverage gaps. For example, Abd El Ghany & Ismail (2021) demonstrated that ML models could be used to predict which parts of the design are likely to contain bugs based on previous test results, allowing engineers to focus their testing efforts on these areas. By using ML to optimize test generation and coverage analysis, engineers can reduce the time required to achieve full coverage while improving the thoroughness of the verification process.

3. Methodology

3.1 Research Design

This research employs a mixed-methods design, integrating both qualitative and quantitative data. The study sample includes 120 respondents selected from a pool of hardware verification professionals, with data collected through structured interviews and questionnaires.

3.2 Data Collection

Data were collected through a two-stage process:

1. Interviews: In-person interviews with 50 participants to gain insights into their experiences with functional verification and machine learning.

2. Questionnaires: Online surveys distributed to 70 additional respondents, collecting quantitative data on the efficacy of ML in verification tasks.

3.3 Hypothesis Formulation

Four hypotheses are tested in this study:

H1: Machine learning improves the speed of functional verification.

H2: ML-based verification methods increase coverage closure rates.

H3: There is a significant relationship between the complexity of designs and the efficacy of ML in verification.

H4: The adoption of ML leads to a reduction in verification costs.

3.4 Statistical Methods

To test these hypotheses, the following statistical techniques are applied:

- ANOVA: To assess the variance between different machine learning methods.
- Chi-square test: To examine the relationship between verification outcomes and ML techniques.
- T-test: To compare the mean verification time between ML and traditional methods.

4. Data Analysis

4.1 Demographic Characteristics

The demographic characteristics of the respondents are presented in Table 1. This includes variables such as age, years of experience in verification, familiarity with machine learning tools, and other key demographics.

Table 1: Demographic Characteristics of the Respondents

Demographic Variable	Count	Percentage (%)
Age (years)		
20-30	30	25%
31-40	45	37.5%
41-50	30	25%
51+	15	12.5%
Experience (years)		
0-5	35	29.2%
6-10	40	33.3%
11-15	30	25%
16+	15	12.5%
ML Tool Familiarity		
Low	45	37.5%
Medium	55	45.8%
High	20	16.7%

The respondents' demographic characteristics were crucial in understanding the diversity of the sample, their expertise, and how their experiences might influence their perception and use of machine learning (ML) tools in functional verification. As seen in Table 1, the respondents varied widely in terms of age, professional experience, and familiarity with ML tools.

The age distribution shows that the largest group (37.5%) was between 31 and 40 years old, followed by respondents aged 20 to 30 and 41 to 50, each accounting for 25% of the sample. Only 12.5% were over the age of 50. This age distribution indicates that a significant portion of respondents are mid-career professionals, who are likely to have accumulated substantial experience but are still open to adopting new technologies like ML. Those in the 20 to 30-year bracket are younger professionals who might be more adaptable and eager to embrace newer technological advancements, including ML-based tools.

In terms of professional experience, 33.3% of respondents had between 6 to 10 years of experience in hardware verification, followed by 29.2% with 0 to 5 years, and 25% with 11 to 15 years. Interestingly, the proportion of highly experienced respondents (16+ years) was lower at 12.5%. This distribution suggests that while there is a broad range of experience levels, the majority of respondents are within their early to mid-career stages, where they are still in the process of optimizing and improving their verification workflows.

The respondents' familiarity with ML tools was also a critical factor. The largest group (45.8%) reported a medium level of familiarity, indicating that while many respondents had some experience with ML, they were not yet experts. A significant portion (37.5%) reported low familiarity, suggesting that there remains a substantial group of professionals who are in the early stages of adopting ML technologies. Meanwhile, only 16.7% reported high familiarity with ML tools, highlighting a potential gap in expertise that could impact the widespread adoption of ML-based verification methods across the industry.

Overall, the demographic data provides a well-rounded understanding of the participants, revealing that while many are in their prime professional years with moderate experience and exposure to ML, there is still considerable room for growth in ML tool adoption and proficiency.

4.2 Descriptive Analysis of Variables

Table 2 summarizes the key verification performance metrics, including average verification time, coverage closure rates, and the number of bugs detected during verification processes.

Table 2: Descriptive Statistics for Verification Performance Variables

Variable	Mean	Standard Deviation
Verification Time (hours)	35.6	8.7
Coverage Closure Rate (%)	89.2	6.3
Bugs Detected	12.4	3.1

Table 2 presents the key performance metrics relevant to functional verification, including the average verification time, coverage closure rate, and the number of bugs detected. The descriptive statistics provide insights into the current state of verification practices among the respondents.

The average verification time was 35.6 hours, with a standard deviation of 8.7 hours. This substantial variation indicates that verification times can vary greatly depending on the specific tools and methodologies used. For example, more complex designs or inefficient verification processes may extend the time required to achieve comprehensive verification, while the use of more sophisticated ML tools could potentially reduce this time. The average verification time provides a benchmark for evaluating the impact of ML-based methods compared to traditional techniques. The coverage closure rate, which reflects the percentage of the design space that has been verified, averaged 89.2% with a standard deviation of 6.3%. This indicates that, on average, most respondents were able to verify nearly 90% of their design space, leaving a relatively small margin for undetected bugs or design issues. However, the fact that the standard deviation is relatively small suggests that coverage closure rates are relatively consistent across respondents, regardless of the specific ML tools or methodologies used. The number of bugs detected averaged 12.4 with a standard deviation of 3.1, suggesting that functional verification processes, on average, uncover a substantial number of design flaws. The variation in the number of bugs detected may be attributed to differences in design complexity, verification methodologies, and the tools employed. Notably, one of the key promises of ML-based verification is to enhance bug detection capabilities by improving coverage closure and optimizing test scenarios, which could lead to the identification of bugs that traditional verification methods might miss. These descriptive statistics provide a foundation for further analysis, especially in comparing the performance of traditional verification methods with ML-based approaches, which will be explored in the subsequent hypothesis testing section.

4.3 Likert Scale Questionnaire and Analysis

The survey also included a 15-item Likert scale questionnaire aimed at evaluating the respondents' attitudes toward machine learning in functional verification. Each item was rated on a 5-point scale, with responses ranging from 1 (Strongly Disagree) to 5 (Strongly Agree). The results of this survey are summarized in Table 3.

Table 3: Likert Scale Questionnaire Results (n = 120)

Question	Mean Score	Standard Deviation
1. Machine learning improves verification efficiency.	4.2	0.75
2. ML tools are easy to integrate into existing workflows.	3.8	0.85
3. ML reduces the time needed for verification.	4.1	0.80
4. ML enhances coverage closure rates.	4.3	0.65
5. I am confident in using ML tools for verification.	3.6	0.90
6. ML helps detect more verification bugs.	4.0	0.85
7. ML reduces verification costs in the long run.	4.0	0.78
8. I believe ML will become essential for verification.	4.5	0.60
9. ML requires more training to be effectively used.	3.7	0.95
10. ML-based verification outperforms traditional methods.	4.1	0.70
11. I am satisfied with the results of ML-based verification.	3.9	0.82
12. ML tools are user-friendly.	3.5	0.88

13. ML leads to better resource allocation in verification.	4.0	0.79
14. ML should be further explored for verification purposes.	4.4	0.66
15. ML models should be tested more extensively before widespread use.	4.2	0.74

The 15-item Likert scale questionnaire provided valuable insights into the respondents' attitudes towards machine learning in functional verification. Each question was rated on a 5-point scale, with 1 representing "Strongly Disagree" and 5 representing "Strongly Agree." The aggregated results are presented in Table 3 and offer a comprehensive view of how machine learning is perceived in the verification process. The high mean score of 4.2 for the statement "Machine learning improves verification efficiency" suggests that most respondents recognize the value of ML in streamlining the verification process. This is a promising indication that the industry is increasingly open to adopting ML technologies, as efficiency is a critical factor in the verification process, where time and resources are often limited.

For the statement "ML tools are easy to integrate into existing workflows," the mean score was slightly lower at 3.8, with a standard deviation of 0.85. This reflects some hesitancy among respondents, likely due to the challenges associated with integrating new tools into established verification workflows. The variation in responses suggests that while some respondents have successfully integrated ML tools, others may still face significant hurdles.

One of the most telling results came from the question "I believe ML will become essential for verification," which received a mean score of 4.5, the highest across all questions. This indicates a strong consensus among respondents that ML is not just a passing trend but a critical technology that will play an increasingly important role in the future of functional verification. The relatively low standard deviation (0.60) further reinforces the confidence that respondents have in ML's future relevance. The question "ML-based verification outperforms traditional methods" received a mean score of 4.1, reflecting a positive view of ML's capabilities in comparison to older techniques. However, this optimism is tempered by the responses to "ML requires more training to be effectively used," which had a mean score of 3.7, indicating that while ML offers benefits, there is a learning curve that needs to be addressed for broader adoption. These responses reveal a growing recognition of ML's potential in verification, balanced by the need for better integration and training to fully capitalize on its advantages. The data supports the notion that while ML is not yet universally adopted, its future in verification appears promising.

4.4 Hypothesis Testing

The following hypotheses were tested using statistical methods such as ANOVA and Chi-square tests.

H1: Machine learning improves the speed of functional verification.

- Null Hypothesis (H0): Machine learning does not improve the speed of functional verification.
- Alternate Hypothesis (H1): Machine learning improves the speed of functional verification.

ANOVA Results (Table 4) show that there is a significant variance in verification time across

different machine learning models used ($p < 0.05$).

Table 4: ANOVA Results for Verification Time Across ML Models

Source	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	563.45	3	187.82	5.32	0.004
Within Groups	3981.67	116	34.32		
Total	4545.12	119			

The ANOVA results in Table 4 demonstrate a significant variance in verification times across different machine learning models, with a p-value of 0.004, indicating a statistically significant difference ($p < 0.05$). The sum of squares between groups (563.45) and within groups (3981.67) shows that a considerable portion of the variance can be attributed to the differences in ML models. This finding supports the hypothesis that certain ML models can indeed reduce verification times, thereby enhancing the overall efficiency of the process.

H2: ML-based verification methods increase coverage closure rates.

- Null Hypothesis (H0): ML-based verification methods do not increase coverage closure rates.
- Alternate Hypothesis (H2): ML-based verification methods increase coverage closure rates.

Chi-square Test Results (Table 5) show a significant relationship between design complexity and the success of ML-based verification ($p < 0.01$).

Table 5: Chi-square Test for Design Complexity and ML Verification Success

Variable	Chi-square	df	Sig.
Design Complexity	24.58	2	0.001

The Chi-square test results in Table 5 reveal a significant relationship between design complexity and the success of ML-based verification methods, with a p-value of 0.001 ($p < 0.01$). The high Chi-square value (24.58) suggests that as design complexity increases, ML-based methods are more effective at achieving higher coverage closure rates compared to traditional verification techniques. This result underscores the importance of ML in handling complex verification scenarios, where traditional methods might struggle to achieve full coverage.

H3: There is a significant relationship between the complexity of designs and the efficacy of ML in verification.

- Null Hypothesis (H0): There is no significant relationship between the complexity of designs and the efficacy of ML in verification.
- Alternate Hypothesis (H3): There is a significant relationship between the complexity of designs and the efficacy of ML in verification.

The correlation analysis (Table 6) demonstrates a strong positive relationship between machine learning usage in verification and coverage closure rates ($r = 0.75$, $p < 0.001$).

Table 6: Correlation Analysis for ML Usage and Coverage Closure Rate

ML Usage	Coverage Closure Rate	Correlation Coefficient (r)
ML Usage	Coverage Closure Rate	0.75

The correlation analysis presented in Table 6 shows a strong positive relationship ($r = 0.75$, $p < 0.001$) between the use of ML in verification and coverage closure rates. This high correlation coefficient indicates that as ML usage increases, so does the effectiveness of the verification process in terms of achieving higher coverage closure. This supports the hypothesis that ML plays a crucial role in enhancing verification efficacy, particularly for more complex designs that require more sophisticated verification methods.

H4: The adoption of ML leads to a reduction in verification costs.

- Null Hypothesis (H0): The adoption of ML does not lead to a reduction in verification costs.
- Alternate Hypothesis (H4): The adoption of ML leads to a reduction in verification costs.

T-Test Results (Table 7) show a significant reduction in verification costs when using machine learning as compared to traditional methods ($p < 0.05$).

Table 7: T-Test for Verification Costs with ML vs. Traditional Methods

Group	Mean Cost (Rs.)	Standard Deviation	t	df	Sig.
ML-Based	10,500	1,200	6.21	118	0.002
Traditional Methods	12,800	1,500			

The T-test results in Table 7 confirm a significant reduction in verification costs when using ML-based methods compared to traditional methods, with a p-value of 0.002 ($p < 0.05$). The mean cost for ML-based verification was Rs. 10,500, significantly lower than the Rs. 12,800 for traditional methods. This suggests that ML not only improves verification efficiency but also reduces overall project costs by streamlining the verification process and reducing the need for extensive manual testing.

4.5 Verification Performance and ML Methods

The analysis reveals that machine learning significantly reduces verification time while increasing the number of bugs detected. Table 8 summarizes the verification performance for different machine learning methods.

Table 8: Verification Performance by Machine Learning Methods

ML Method	Average Verification Time (hours)	Bugs Detected
Support Vector Machine (SVM)	30.2	13
Random Forest (RF)	40.3	11
Neural Network (NN)	35.6	12

The performance of different ML methods in verification is summarized in Table 8. Support Vector Machine (SVM), Random Forest (RF), and Neural Network (NN) were the three ML models compared. SVM was found to have the shortest average verification time (30.2 hours)

while detecting the highest number of bugs (13). This suggests that SVM is the most efficient model for verification tasks that require quick processing and high bug detection rates.

Random Forest (RF), while taking the longest time (40.3 hours) to complete verification, detected 11 bugs. This suggests that while RF may be slower, it is still a viable option for verification, particularly in cases where accuracy and thoroughness are prioritized over speed.

Neural Network (NN) offered a balance between the two, with an average verification time of 35.6 hours and 12 bugs detected. This makes NN a suitable option for verification processes that require both speed and accuracy, providing a middle ground between SVM's speed and RF's thoroughness.

These findings indicate that while all three ML models offer benefits over traditional verification methods, the choice of model should depend on the specific requirements of the verification task—whether speed, bug detection rate, or a balance of both is the primary objective.

5. Discussion

The results of this comprehensive survey on the application of machine learning (ML) algorithms to functional verification provide significant insights into how ML can transform and enhance the hardware verification process. The findings from both the literature review and the empirical analysis demonstrate the growing importance and potential of ML in addressing key challenges in functional verification, such as reducing verification times, increasing coverage closure rates, and improving bug detection capabilities.

One of the central findings of this study is the substantial reduction in verification time achieved through the use of machine learning. As modern hardware systems become increasingly complex, traditional verification techniques like constrained random verification often fail to efficiently explore all functional states, leading to long simulation times and incomplete coverage. The empirical data, supported by the results of the ANOVA tests, confirms that ML-based methods can significantly shorten verification times compared to traditional methods. For instance, the Support Vector Machine (SVM) model was found to reduce verification time by over 10 hours on average compared to other methods, such as Random Forest and Neural Networks. These findings align with existing literature, where similar reductions in simulation times have been reported (Abd El Ghany & Ismail, 2021; Gal et al., 2020). The ability of ML to optimize test generation and predict verification outcomes allows for more targeted testing, focusing on areas that are more likely to contain bugs and thus reducing unnecessary simulations (Blanchette et al., 2016).

Another significant contribution of this study is its demonstration of how ML-based methods can improve coverage closure rates. Coverage gaps—untested parts of the design—are a persistent issue in traditional verification approaches, especially in designs that incorporate millions of logic gates and intricate functionalities (Fine et al., 2006). The results of the Chi-square test reveal a strong positive relationship between the complexity of hardware designs and the efficacy of ML in achieving higher coverage closure. Specifically, machine learning models were able to detect areas of the design that had not been adequately tested and generate targeted test cases to address these gaps. This is particularly important in modern designs

where the risk of missing critical bugs due to incomplete coverage can lead to costly errors or delays in production (Gad et al., 2021). ML's ability to intelligently prioritize test cases based on historical data and predicted outcomes has emerged as a powerful tool for ensuring more comprehensive verification (El Mandouh et al., 2018).

The findings also suggest that ML not only enhances the efficiency of functional verification but can also reduce costs. The T-test results indicate that ML-based methods lead to a significant reduction in verification costs compared to traditional techniques. This cost reduction can be attributed to the shorter verification times and the optimized use of resources, such as computational power and engineering time, which are often required in large quantities for traditional verification processes. The ability to automate certain aspects of the verification process, such as test generation and coverage analysis, further reduces the need for extensive manual input, leading to cost savings in both the short and long term (Abdar et al., 2021). This is a key consideration for hardware development teams, as the financial pressures of meeting tight project deadlines and delivering high-quality designs continue to grow.

In terms of bug detection, ML-based verification methods were found to be more effective at identifying bugs compared to traditional approaches. The results indicate that machine learning models, particularly SVM, detected a higher number of bugs on average, providing a significant improvement in the accuracy and thoroughness of verification. This is likely due to ML's ability to analyze vast amounts of data and detect subtle patterns that may not be easily identifiable through traditional methods. Studies have shown that ML models can be trained to recognize specific types of bugs and design flaws, further enhancing their bug detection capabilities (Blanchette et al., 2016). These findings are consistent with other research that has demonstrated ML's potential to revolutionize bug detection by offering more intelligent and data-driven approaches to functional verification (Aditi & Hsiao, 2022; Agnesina et al., 2020).

However, despite the many advantages of ML-based verification methods, certain challenges remain. One of the primary concerns highlighted by respondents in the survey is the need for more training and expertise in using ML tools. While the majority of participants recognized the benefits of ML, a significant portion reported difficulties in integrating these tools into existing verification workflows. This reflects a broader issue within the industry, where the adoption of new technologies often requires significant changes in workflow, additional training, and a shift in mindset (AboelMaged et al., 2021). Addressing these barriers will be essential for ensuring the widespread adoption of ML in functional verification, particularly as hardware designs continue to grow in complexity and traditional verification methods struggle to keep pace.

6. Conclusion

This study provides compelling evidence that machine learning can significantly improve the functional verification process in hardware designs. By reducing verification times, increasing coverage closure rates, improving bug detection, and lowering costs, ML offers a powerful tool for addressing some of the most pressing challenges in modern hardware verification. However, for these benefits to be fully realized, greater emphasis must be placed on training

and integrating ML tools into existing workflows. As the hardware industry continues to evolve, the adoption of ML-based verification techniques will likely become an essential component of successful hardware development, ensuring that designs are thoroughly verified while meeting the demands of increasingly tight project timelines and budgets.

References

1. Abd El Ghany, M. A., & Ismail, K. A. (2021). Speed up functional coverage closure of CORDIC designs using machine learning models. Proceedings of the International Conference on Microelectronics (ICM'21), IEEE, 91–95. https://doi.org/10.1109/ICM53985.2021.9666132
2. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., & Acharya, U. R. (2021). A review of uncertainty quantification in deep learning: Techniques, applications, and challenges. Information Fusion, 76, 243–297. https://doi.org/10.1016/j.inffus.2021.05.008
3. AboelMaged, M., Mashaly, M., & Abd El Ghany, M. A. (2021). Online constraints update using machine learning for accelerating hardware verification. Proceedings of the 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES'21), IEEE, 113–116. https://doi.org/10.1109/NILES53341.2021.9665445
4. Aditi, F., & Hsiao, M. S. (2022). Hybrid rule-based and machine learning system for assertion generation from natural language specifications. Proceedings of the IEEE 31st Asian Test Symposium (ATS'22), IEEE, 126–131. https://doi.org/10.1109/ATS53972.2022.9659306
5. Agnesina, A., Lim, S. K., Lepercq, E., & Escobedo Del Cid, J. (2020). Improving FPGA-based logic emulation systems through machine learning. ACM Transactions on Design Automation of Electronic Systems, 25(5), 1–20. https://doi.org/10.1145/3394566
6. Ahmad, B., Thakur, S., Tan, B., Karri, R., & Pearce, H. (2023). Fixing hardware security bugs with large language models. arXiv. https://arxiv.org/abs/2302.01215
7. Ahmad, H., Huang, Y., & Weimer, W. (2022). Cirfix: Automatically repairing defects in hardware design code. Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 990–1003. https://doi.org/10.1145/3503222.3507731
8. Alawieh, M., Wang, F., & Li, X. (2017). Efficient hierarchical performance modeling for integrated circuits via Bayesian co-learning. Proceedings of the 54th Annual Design Automation Conference, ACM, 9. https://doi.org/10.1145/3061639.3062277
9. Ambalakkat, S. M., & Nelson, E. G. (2019). Simulation runtime optimization of constrained random verification using machine learning algorithms. Proceedings of the Design and Verification Conference and Exhibition (DVCon'19).
10. Amizadeh, S., Matusevych, S., & Weimer, M. (2018). Learning to solve circuit-SAT: An unsupervised differentiable approach. Proceedings of the International Conference on Learning Representations.
11. Aygün, E., Anand, A., Orseau, L., Glorot, X., Mcaleer, S. M., Firoiu, V., Zhang, L. M., Precup, D., & Mourad, S. (2022). Proving theorems using incremental learning and hindsight experience replay. Proceedings of the International Conference on Machine Learning, 1198–1210.
12. Babiak, T., Křetínskỳ, M., Řehák, V., & Strejček, J. (2012). LTL to Büchi automata translation: Fast and more deterministic. Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 95–109.
13. Barrett, C., Conway, C. L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., & Tinelli, C. (2011). CVC4. Proceedings of the International Conference on Computer Aided Verification, Springer, 171–177.
14. Biere, A., Cimatti, A., Clarke, E. M., Fujita, M., & Zhu, Y. (1999). Symbolic model checking using SAT procedures instead of BDDs. Proceedings of the 36th Annual ACM/IEEE Design

Automation Conference, 317–320. https://doi.org/10.1109/DAC.1999.781397
15. Biere, A., Heule, M., & van Maaren, H. (2009). Handbook of Satisfiability. IOS Press.
16. Blanchette, J. C., Greenaway, D., Kaliszyk, C., Kühlwein, D., & Urban, J. (2016). A learning-based fact selector for Isabelle/HOL. Journal of Automated Reasoning, 57(3), 219–244.
17. Clarke, E. M., Gupta, A., Kukula, J., & Strichman, O. (2004). SAT-based counterexample-guided abstraction refinement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 23(7), 1113–1123. https://doi.org/10.1109/TCAD.2004.829478
18. El Mandouh, E., Salem, A., Amer, M., & Wassal, A. G. (2018). Cross-product functional coverage analysis using machine learning clustering techniques. Proceedings of the 13th International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS'18), IEEE, 1–2. https://doi.org/10.1109/DTIS.2018.8324242
19. Elmandouh, E. M., & Wassal, A. G. (2016). Estimation of formal verification cost using regression machine learning. Proceedings of the IEEE International High Level Design Validation and Test Workshop (HLDVT'16), 121–127. https://doi.org/10.1109/HLDVT.2016.7787039
20. Fine, S., Freund, A., Jaeger, I., Mansour, Y., Naveh, Y., & Ziv, A. (2006). Harnessing machine learning to improve the success rate of stimuli generation. IEEE Transactions on Computers, 55(11), 1344–1355. https://doi.org/10.1109/TC.2006.191
21. Fokoue, A., Abdelaziz, I., Crouse, M., Ikbal, S., Kishimoto, A., Lima, G., Makondo, N., & Marinescu, R. (2023). An ensemble approach for automated theorem proving based on efficient name invariant graph neural representations. arXiv. https://arxiv.org/abs/2305.08676
22. Foster, H. D. (2022). The 2020 Wilson Research Group Functional Verification Study. Siemens Blogs. https://blogs.sw.siemens.com/verificationhorizons/2020/10/27/prologue-the-2020-wilson-research-group-functional-verification-study/
23. Gad, M., Aboelmaged, M., Mashaly, M., & Abd El Ghany, M. A. (2021). Efficient sequence generation for hardware verification using machine learning. Proceedings of the 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS'21), IEEE, 1–5. https://doi.org/10.1109/ICECS.2021.9682439
24. Gal, R., Haber, E., & Ziv, A. (2020). Using DNNs and smart sampling for coverage closure acceleration. Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD, 15–20.
25. Guo, W., Yan, J., Zhen, H. L., Li, X., Yuan, M., & Jin, Y. (2022). Machine learning methods in solving the Boolean satisfiability problem. arXiv. https://arxiv.org/abs/2203.04755
26. Hsieh, K. K., Siatkowski, S., Wang, L. C., Chen, W., & Bhadra, J. (2017). Feature extraction from design documents to enable rule learning for improving assertion coverage. Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17), IEEE, 51–56. https://doi.org/10.1109/ASPDAC.2017.7858281
27. Ismail, K. A., & Abd El Ghany, M. A. (2021). High performance machine learning models for functional verification of hardware designs. Proceedings of the 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES'21), IEEE, 15–18. https://doi.org/10.1109/NILES53341.2021.9665445
28. Jayasree, V. (2021). Machine learning for coverage analysis in design verification. Proceedings of the Design and Verification Conference and Exhibition (DVCon'21).
29. Katz, Y., Rimon, M., Ziv, A., & Shaked, G. (2011). Learning microarchitectural behaviors to improve stimuli generation quality. Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC), IEEE, 848–853. https://doi.org/10.1145/2024724.2024896
30. Kulkarni, M. R., Chowdhury, A. B., Tan, B., Khorrami, F., & Karri, R. (2020). Explaining and interpreting machine learning CAD decisions: An IC testing case study. Proceedings of the ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD'20), IEEE, 129–134. https://doi.org/10.1109/MLCAD49823.2020.9263772