A Pedagogical Framework for Teaching Verilog-Based Digital System Design Using Inquiry-Based Learning

Audre Arlene A[1], Dr. Jagadeesh Basavaiah[2], Dr. Chandrashekar Mohan Patil[3], Dr. Geethashree A[4]
[1,2,3,4] Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering
[1]audre.arlene@vvce.ac.in,
[2]jagadeesh.b@vvce.ac.in ,
[3]patilcm@vvce.ac.in,
[4]geethashree.a@vvce.ac.in

***Abstract*—This study investigates the application of Inquiry-Based Learning (IBL) in the context of Digital System Design (DSD) with Verilog to address limitations of traditional teaching approaches. Conventional instruction in hardware description language courses emphasize syntax and procedural knowledge but often neglect critical thinking and problem-solving skills essential for engineering practice. To overcome these challenges, an IBL framework was designed and implemented, wherein students engaged in guided exploration, hypothesis generation, and iterative project development. The paper details the instructional design, deployment strategies, and evaluation of IBL-based modules, demonstrating that the approach enhances both conceptual understanding and practical competence in Verilog. Comparative observations revealed improvements in student engagement, design skills, and debugging strategies, with measurable gains in creativity and independent inquiry. The results suggest that incorporating IBL into DSD curricula not only strengthens technical expertise but also cultivates the higher-order thinking skills necessary for modern engineering challenges.**

***Keywords*—Inquiry-Based Learning, Digital System Design, Verilog, Engineering Pedagogy, Problem-Solving, Student Engagement.**

ICTIEE* Track— *Innovative Pedagogies and Active Learning
ICTIEE Sub-Track—Inquiry-Based Learning in Fostering Curiosity and Critical Thinking among GenZ.

I. INTRODUCTION

THIS he rapid advancement of digital technologies and the increasing complexity of computing systems have underscored the need for innovative teaching and learning methodologies in engineering education. Digital System Design (DSD), with hardware description languages such as Verilog, plays a pivotal role in engineering education. It helps students design, simulate, and implement digital circuits and systems effectively. However, traditional pedagogical approaches in teaching DSD often emphasize rote learning and step-by-step instruction, which may limit students' problem-solving skills, critical thinking, and creativity in real-world applications (M. Prince, 2004).

Inquiry-Based Learning (IBL) has emerged as a transformative pedagogical strategy that shifts the focus from passive knowledge acquisition to active knowledge construction. Through a didactic process that involves engaging students to question, explore other solutions and conduct practical experiments, IBL is able to create deeper knowledge and a better knowledge retention in the long term (J. Dewey, 1938). In terms of engineering education, IBL promotes constructivist learning theories, which lays out in an active construction of knowledge by learners through exploration and reflection instead of imparting learning to the learners as a passive recipient of information (J. W. Creswell, 2012).
Incorporation of IBL in the teaching of Digital System Design with Verilog will be a rare chance to engage and make the students competent. Unlike typical lecture-based course delivery formats, IBL gives students time to frame design problems and develop hypotheses, test their reasoning using simulation tools, and adjust designs based on observed results. This design methodology not only reflects the engineering methods in hardware design and verification, but it also empowers students with skills that are vital in the engineering field, such as problem decomposition, debugging, and design optimizations (C. Chin et al., 2008). In addition, IBL fosters a learning culture of teamwork where students work together in groups to draft, simulate, and test digital systems, thus translating to teamwork and communication skills that are so desirable in semiconductor and VLSI companies.

Although it has a potential role to play, the topic of applying IBL to Digital System Design suing Verilog is not widely represented in existing literature, where majority of the literature pertains to traditional instructional models only. Thus, this research aims to explore the usefulness of IBL approaches to teach Verilog-based structural design of digital systems with a focus on its effect on the student performance, engagement, and development of higher-order thinking skills. With the expanding field of knowledge concerning active learning pedagogy in engineering education, the results of the present study are likely to become useful insights in further

Audre Arlene A
Vidyavardhaka College of Engineering Mysore
audre.arlene@vvce.ac.in

modernizing DSD pedagogy and will be useful input to curriculum developers, instructors, and decision-makers.

II. RELATED WORK

In this section, the current methodologies available in the field of IBL in engineering education was summarized, with a focus on its applications to digital system design using Verilog. IBL has gained considerable importance in the field of engineering education because it encourages critical thinking, problem-solving, and self-directed learning.

For instance, Prince and Felder (R. M. Felder et al., 2006) declared that student engagement and conceptual retention are higher through active and inquiry-based methods compared to the common ones. In the same context, Savery (J. Savery, 2006) emphasized that IBL fosters high-level thinking where students are encouraged to ask questions, research the answers, and give reflections. When applied to engineering, (Hmelo-Silver et al., 2006) have found that students learn more about design principles through IBL as they are placed in real-world problem-solving situations.

Conducting a study on the digital design education in particular, (Alnajjar et al., 2018) demonstrated the positive effect of the blending of educational inquiry-based labs and Verilog on the ability of the students to move the theory of logic design into hardware description. The research found that the experience in the form of inquiry-based labs led to the increased problem-solving abilities of the students as compared to the subjects of the control group. Similar results were found when inquiry-based instruction was applied to digital logic design education courses by (Y. Chang et al., 2018), and they indicated greater knowledge retention of such concepts as multiplexers, decoders, and sequential circuits.

(S. Kumar et al., 2019) evaluated an inquiry-based approach to learning hardware description languages. Their study found that IBL improved students' motivation, problem-solving skills, debugging ability and creativity. Similarly, (Chen et al., 2019)demonstrated that project-based FPGA design coursework encouraged the pragmatic implementation skills development and increased collaboration among students.

Courses centering on Verilog are not the only contexts where inquiry-based frameworks have been found useful. Specifically, all the technological benefits with regard to the learning process, exemplified by (Yadav et al., 2020), that IBL integration into computer engineering courses resulted in a considerable rise in problem formulation and problem solution evaluation skills of the students. Likewise, (Freeman et al., 2014) presented a meta-analysis of 225 studies conducted in STEM and confirmed that inquiry-based and active learning techniques had a significant positive impact on learning outcomes and decreased the failure rates.

Comparative studies also depict the efficacy of IBL. During the study by (Strobel et al., 2009), the inquiry-based approach produced higher long-term retention of knowledge in comparison with the traditional approaches. (L Li et al., 2020) have shown that in the digital systems area, inclusion of inquiry-based simulations aids students in developing a solid conceptual framework of both the syntax and semantics of Verilog. (Banerjee et al., 2021) also demonstrated that skill in design validation, teamwork, and analytical reasoning were enhanced with inquiry-based Verilog laboratories.

More recent investigations also extend these findings. (Park et al., 2021) noted that when inquiry-based strategies were applied to sequential circuit design, students demonstrated improved error detection and correction skills. Likewise, (Dasgupta et al., 2021) emphasized that inquiry-based project assignments in VLSI design courses promoted independent research and creativity in problem-solving. (Rahman et al., 2022) demonstrated that integrating IBL into Verilog-based design courses facilitated not only technical competence but also communication and presentation skills, preparing students more holistically for industry demands.

Overall, the literature suggests that IBL is an effective paradigm for enhancing conceptual understanding, practical problem-solving, and critical thinking in digital system design using Verilog. Previous research has investigated the use of inquiry-based learning (IBL) methods for engineering and computing education. In particular, (Park et al., 2021) are one of the few studies that have examined the systematic use of these methods within the context of Verilog-based digital system design. Most prior research has focused on general conceptual development or problem-based learning within the context of engineering, with little attention being paid to the unique challenges (e.g., concurrency, timing behaviour, simulation interpretation, and debugging) faced by those using Verilog.

The unique characteristics of Verilog create a need for IBL because students are required to refine their designs based on structured, iterative feedback. Thus, there is an immediate pedagogical benefit to establishing a Verilog-based IBL framework that would foster deeper understanding of HDL concepts and enhance the simulator's ability to interpret simulations, ultimately building stronger and more versatile digital design skills.

III. METHODOLOGY

The methodology adopted in this study is structured to integrate IBL principles into the teaching of Digital System Design using Verilog. It provides a systematic pathway for planning, implementing, and evaluating the instructional framework. The approach is designed not only to enhance conceptual understanding but also to promote higher-order thinking, problem-solving, and teamwork among students. To achieve this, the methodology is divided into four interconnected components: the IBL framework, implementation strategy, design of inquiry-based tasks, and assessment and feedback mechanism.

A. Inquiry-Based Learning Framework

IBL offers a student-centred model in which learning is curiosity-driven, discovery-based, and problem-centred as opposed to directed methods. IBL fits naturally into the design-test-debug cycle through which engineers work and so is well suited to the training of both theoretical and practical skills.

The framework is designed around the problem statements, which increases the complexity. Each issue needed students to be involved in a project-based learning that was supported by

Verilog-based experiments. The most basic tasks included the design of basic logic gates along with their verification using Verilog simulation, whereas more complex tasks included the design of Arithmetic logic units (ALUs), state machines, and memory module designs. Such scaffolded development enabled the learners to use previous knowledge in efficient inquiry.

To facilitate this process, students were prompted to ask questions like: What is the best way to go about implementing this logic circuit in Verilog? or How do the resource uses vary when going between behavioral and structural models? By posing learning questions in this more open-ended manner, students became exposed to a higher level of critical thinking and problem-solving than would have been required to memorize syntax or a pre-set series of design steps.

Also, mini projects, which resembled real-life design situations, were incorporated in the IBL framework. In these projects, creativity and exploration are encouraged, which allowed students to explore ideas without penalty for early mistakes, leading students to make a hypothesis about how it could be implemented and test the design using simulation platforms like ModelSim or Xilinx, or Vivado, and then make iterative improvements until it works. By defining students as problem-solvers instead of passive receivers of knowledge, the framework encouraged conceptual understanding and practical abilities, which proved useful to engineers of modern digital systems.

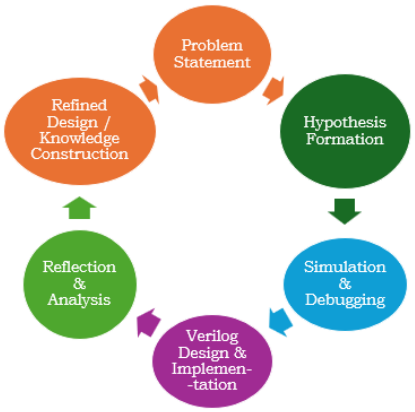

Fig.1. Inquiry-Based Learning Framework for Digital System Design using Verilog

The phases of the inquiry-based procedure were logically arranged with the design-test-debug loop of digital system design. The proposed IBL framework of Verilog-based courses is depicted in Fig.1. The cycle starts with problem statements that would invite students to pose pertinent design queries. This is superseded by the generation of hypotheses and design ideas, which are realized through Verilog. Simulation and verification using tools like ModelSim or Xilinx, or Vivado allow instant verification and correction. The cycle then culminates in a knowledge construction and the subsequent output of tangible outcomes with respect to the project and hence enhancing not only a theoretical grasp but also the practical skillset instilled.

In the early modules, scaffolding was used to support novice students by using brief instructor-led demonstrations and template examples for Verilog syntax. As the students gained competency in using Verilog, these support elements were diminished, and they began to transition into fully unstructured inquiry-based tasks.

B. Implementation Strategy

The IBL model was precisely aligned with the course outline of Digital System Design using Verilog to shift conventional teaching to exploration, experimentation, and refinement. All syllabus modules were organized based on inquiry-based assignments that prompted students to ask questions, make guesses, elaborate solutions, and test their knowledge using simulation and synthesis. Instead of relying on instructor-provided code, students explored multiple design options. They made justified choices based on simulation outcomes. This shift promoted not only the acquisition of Verilog syntax but also the development of design intuition, debugging strategies, and higher-order problem-solving skills.

The frequent use of structured checkpoints established within each inquiry-based activity enabled students to experience a number of supported opportunities to develop proficiency in the common debugging and syntax challenges that are prevalent in Verilog programming.

Table I presents the mapping from syllabus modules to inquiry-based tasks, example activities, and expected outcomes. Each module intentionally paired two levels of inquiry: a conceptual task that pressed students to interrogate principles (e.g., the role of hardware description languages; the functional implications of blocking versus non-blocking assignments) and an application task that translated those principles into working artifacts (e.g., a traffic-light controller, carry-lookahead adder, or ALU). The combination ensured that foundational reasoning was reinforced while progressively challenging learners to synthesize and extend their knowledge.

TABLE I
MAPPING OF SYLLABUS MODULES TO INQUIRY-BASED TASKS

Module	Key Concepts	inquiry-based Tasks	Example Activities	Expected Outcomes
Module 1: Basic Concepts	Evolution of CAD, HDLs, Verilog syntax, modules, ports	1. Why do we need HDLs instead of schematic-based design? 2. How do port declaration styles influence module connectivity?	1. Create a simple Verilog module and test input/output behavior. 2. Experiment with different port connection rules and analyze simulation results.	Students understand the necessity of HDLs, gain familiarity with syntax, and explore port handling through trial and error.
Module 2: Gate-Level Modeling	Gate primitives, delays (rise/fall/turn – off)	1. How does delay modeling affect circuit behavior in simulations? 2. What differences exist between structural and primitive gate modeling?	1. Implement AND/OR/NOT gates with varying delays; compare timing in waveforms. 2. Build a 2:1 multiplexer using gate-level modeling and verify operation.	Students understand delay modeling, timing behavior, and structural vs. primitive design styles.

Module 3: Dataflow Modeling	Continuous assignments, operators, multiplexer, adder	1. What are the trade-offs between gate-level and dataflow modeling? 2. How do operators impact the efficiency and readability of Verilog code?	1. Implement a 4:1 multiplexer using continuous assignments; test functionality. 2. Design a 4-bit full adder with carry-lookahead using dataflow style.	Students recognize abstraction levels, explore operator usage, and appreciate dataflow efficiency.
Module 4: Behavioral Modeling	Always blocks, blocking vs. non-blocking, loops, and conditional execution	1. How does the choice between blocking and non-blocking assignments affect functionality? 2. What role do loops and conditional statements play in behavioral modeling?	1. Simulate a counter using both blocking and non-blocking assignments; compare outputs. 2. Implement a traffic light controller using loops and case statements.	Students internalize event control, timing behavior, and high-level design strategies.
Module 5: Tasks, Functions, and Synthesis	Reusable modules, synthesis flow, netlist verification	1. How can reusability and modularity be achieved in Verilog? 2. What optimizations are introduced during synthesis compared to RTL coding?	1. Implement arithmetic operations (e.g., comparator, adder) using tasks/functions; reuse them in larger modules. 2. Synthesize a behavioral design (e.g., ALU) and compare the gate-level netlist with the RTL description.	Students develop modular design skills, understand synthesis flow, and verify netlist optimizations.

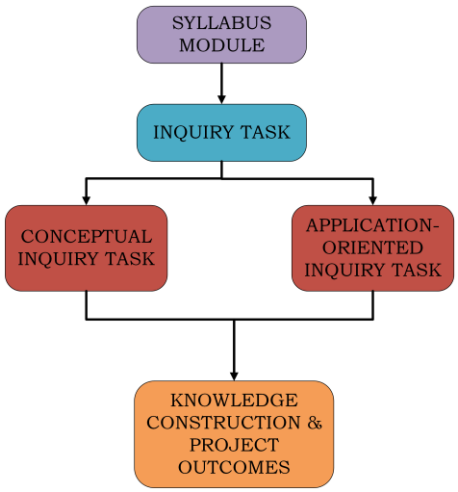

Fig.2. Mapping of Inquiry-based Tasks to Verilog Design Process

The implementation followed a recursive inquiry-based cycle: problem identification → hypothesis generation → Verilog implementation → simulation and testing using simulation tools → reflection and analysis → refinement and debugging → knowledge construction. For example, in gate-level modeling, students predicted how rise/fall delays shape behavior, verified predictions via waveform inspection, and refined structural designs; in behavioral modeling, they contrasted blocking/non-blocking semantics conceptually, then engineered a finite-state-machine controller and resolved race conditions through iterative edits. The end-to-end mapping—from syllabus topics, through dual inquiry-based branches, into validated designs—is summarized in Fig.2, which depicts how conceptual and application inquiries converge into implementation, simulation, refinement, and ultimately knowledge construction and project outcomes.

C. Design of Inquiry Tasks

Building on the syllabus-task mapping presented in Table I, the inquiry-based tasks were structured to gradually increase in complexity and abstraction, reflecting the natural progression of digital system design. The design philosophy emphasized scaffolding—beginning with gate-level modeling, moving through dataflow and behavioral abstractions, and culminating in modular design and synthesis.

Each task was anchored in an inquiry-based question that required students to hypothesize, design, simulate, and refine their solutions. This ensured that the focus was not on reproducing instructor-provided code, but on independent exploration, justification of design choices, and critical reflection.

At the conceptual level, tasks encouraged students to interrogate principles such as propagation delays, abstraction trade-offs, and assignment semantics, while at the application level, students translated these insights into practical artifacts such as multiplexers, adders, counters, finite state machines, and synthesized netlists.

To illustrate this structured progression, Fig.3 depicts the hierarchical design of inquiry-based tasks, organized as a pyramid. At the base, students begin with gate-level modeling to establish foundational timing concepts. They then advance to dataflow abstractions for arithmetic design, followed by behavioral modeling of sequential systems where blocking vs. non-blocking semantics and control logic are interrogated. The upper layers emphasize modular design through tasks and functions, culminating in synthesis and netlist analysis that reflect industry-relevant optimization and verification practices.

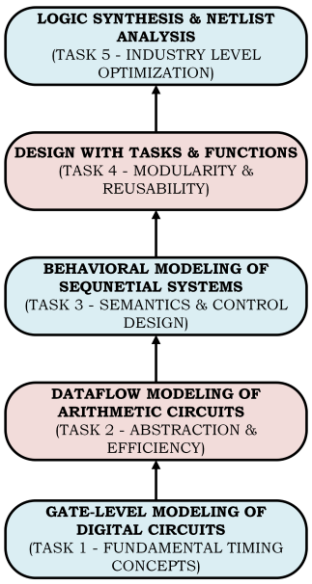

Fig.3. Hierarchical Progression of Inquiry-based Tasks in Digital System Design

This scaffolded design allowed learners to acquire syntactic fluency in Verilog while simultaneously developing deeper insights into digital system design methodologies, debugging

strategies, and higher-order inquiry-based skills.

D. Assessment and Feedback Mechanism

To rigorously evaluate the effectiveness of the IBL approach in Digital System Design using Verilog, a multi-layered assessment and feedback mechanism was adopted. This ensured that both student learning outcomes and instructional strategies were continuously monitored, validated, and refined.

1) Pre-test vs. Post-test Comparison

Student learning was quantitatively assessed through pre-test and post-test evaluations. The pre-test, conducted before IBL implementation, measured baseline knowledge of Verilog syntax, modeling approaches, and logic design principles. After completing the IBL modules, students undertook a post-test of equivalent complexity.

As illustrated in Fig. 4, the comparative analysis revealed substantial improvement in post-test performance, highlighting not only conceptual growth but also enhanced problem-solving skills in Verilog-based design. Standard deviation error bars are included to show variability in student performance.

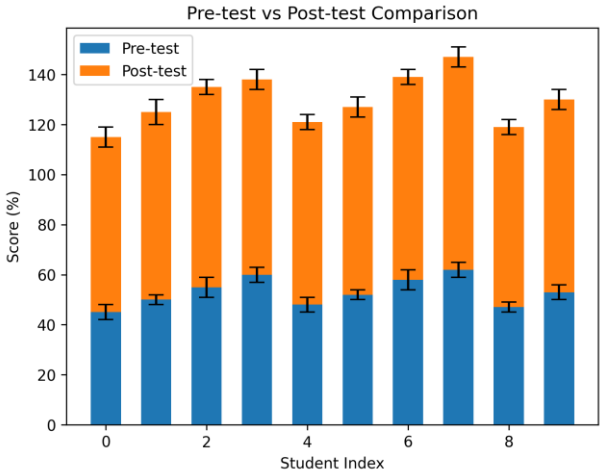

Fig. 4. Pre-test vs. Post-test Comparison.

2) Lab Report Submissions

Following each inquiry-based task, students submitted structured lab reports that included:

- Documentation of hypotheses and experimental setup
- Verilog code snippets with test benches
- Simulation results and waveform analysis
- Reflections on debugging strategies

These reports encouraged self-explanation, accountability, and technical writing, reinforcing critical design documentation practices.

3) Peer Review of Projects

Collaborative learning was fostered through peer review sessions. Each project team evaluated another group's design, providing feedback on:

- Code efficiency and style
- Modularity and reusability of design
- Clarity of simulation outputs
- Creativity and innovation in approach

This mechanism promoted critical thinking, constructive criticism, and teamwork, creating a culture of shared learning and improvement.

4) Rubric-based Grading

To ensure fairness and transparency, a rubric-based grading system was employed. The rubric allocated weights to key performance dimensions, as shown in Table II.

TABLE II
RUBRIC-BASED GRADING DISTRIBUTION

Criterion	Weight	Description
Design Accuracy	30%	Correctness of logic, synthesis, and simulation results
Innovation	20%	Novelty in design approach and optimized coding practices
Teamwork	20%	Collaboration, task division, and peer contributions
Presentation	15%	Clarity of oral/visual presentation
Documentation	15%	Quality of reports, code clarity, and reflection

5) Feedback Collection

Feedback was continuously integrated at multiple levels:

- Student surveys captured perceptions on clarity of inquiry tasks, workload, and learning gains.
- Reflective essays allowed students to articulate challenges and personal growth.
- Instructor feedback on reports and projects provided iterative guidance on coding practices and conceptual understanding.

The integration of these elements is depicted in Fig.5, which presents the assessment and feedback mechanism as a structured flow.

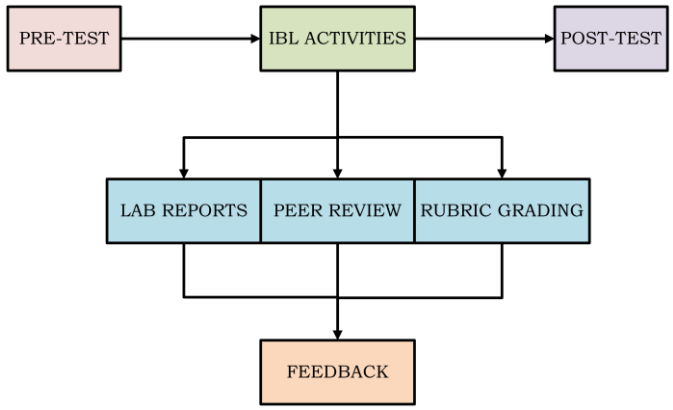

Fig.5. Assessment and Feedback Flowchart.

Furthermore, Fig.6 illustrates the feedback loop between students, instructors, and inquiry-based tasks. Student reflections and surveys informed instructors, who then refined inquiry-based tasks, ensuring a cycle of continuous improvement in teaching and learning.

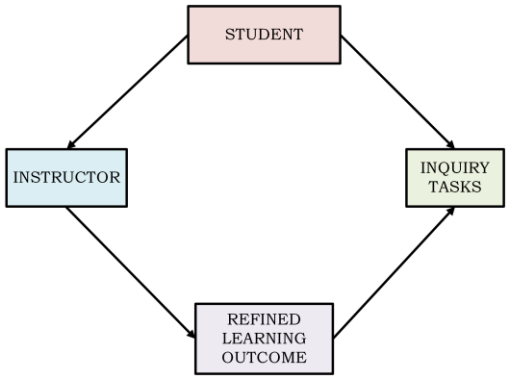

Fig.6. Feedback Loop Mechanism.

A Continuous, Iterative Cycle of Assessment and Instructional Feedback was employed in this study. The participants all completed a Pre-Test as well as an initial inquiries assignment, then received feedback based on the errors made. The inquiries that followed were designed so that they could use the feedback received to progressively improve their Verilog design and debug skills. The reflections on the assignment were used to create instructor responses that were targeted toward the needs of the participant, and the project evaluations matched the competencies that were reinforced throughout the iterations. The linked assessment and feedback processes are shown in Fig.5 and Fig.6 for all four phases of the learning process (development, practice, application and evaluation).

Alignment with Accreditation Outcomes: Accreditations are aligned with and meet expectation for accreditation of ABET student outcomes 1 and 2. The student's capacity to identify, formulate and/or evaluate engineering problems (outcome 1) is enhanced through an emphasis on iterative inquiry tasks. Students' ability to apply engineering design principles to produce solutions satisfying defined requirements (outcome 2) is reinforced by the inclusion of reflective activities and project-based assessments. With SACSCOC competency-based standards being enforced, the continuous improvement cycle enables the supporting of the students to establish ongoing progress.

This study was performed on a group of second-year engineering students taking a Digital System Design course (which all students were required to take). All of the students had had some introductory experience with using logic circuits tarted and had not taken a course in Verilog or simulation using HDL. The study utilized convenience sampling, which may have resulted in selection bias because it was completed at a single school. To limit any threats to the study's internal validity all students received the same course materials, and assessments were given to all students under identical conditions.

The Appendix contains supporting materials such as example inquiry tasks, anonymised student reflections on their experiences, the coding framework for the qualitative analysis, and the pre-test and post-test tools.

IV. Assessment and Feedback Mechanism

The effectiveness of the IBL framework in Digital System Design using Verilog was evaluated through both quantitative performance metrics and qualitative feedback obtained from students and instructors. A total of 63 undergraduate students participated in this study.

A. Metrics of Evaluation

The evaluation considered the following dimensions:

- Student Performance – measured through pre-test vs. post-test scores, lab reports, and final project grades.
- Engagement Levels – assessed through participation in class discussions, teamwork dynamics, and timely submission of reports.
- Innovation in Projects – evaluated based on the novelty of design approaches, efficient Verilog coding practices, and simulation accuracy.

Student Feedback – collected through structured surveys and reflective essays.

Students with difficulty in Verilog syntax and debugging received targeted assistance in form of brief troubleshooting demonstrations and example-based descriptions of the errors made. The short interventions provided students with an opportunity to identify common mistakes and formulate their own ways to troubleshoot issues.

B. Learning Gains: Pre-test vs. Post-test

To assess conceptual learning, a diagnostic pre-test and a summative post-test were administered.

1. Pre-test average score: 52.3%
2. Post-test average score: 76.8%
3. Learning gain: ≈ 24.5 percentage points

This substantial improvement highlights the positive impact of IBL on both conceptual understanding and application skills. Fig.7 illustrates the comparative analysis of pre-test vs. post-test performance. Standard deviation error bars are included to indicate variability in scores.

Normality of differences in scores (pre and post) was assessed as part of the verification of the paired t-test assumptions using the Wilk test. All analysis was conducted using an alpha of 0.05. In addition to demonstrating statistical significance ($p < 0.001$), a large practical effect size was identified with Cohen's d, indicating a strong learning effect was achieved. The construction of 95% confidence intervals around the mean difference provided more evidence of the increase in learning achieved by all of the students involved in the study.

Measurement and Statistical Analysis - Gains in learning were measured using paired T-tests to compare pre and post-test results. Normalcy assumptions were checked and confirmed for how differences in scores behaved. To measure the size of the effect on practical significance (large improvement) Cohen's d was computed as an effect adjustment. As well, descriptive statistics (mean & SD) were presented for all achievements to give a clearer view of improvements in student learning activity.

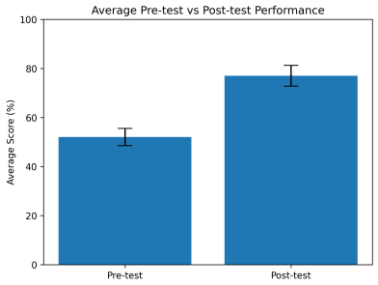

Fig.7. Average Pre-test vs Post-test Performance.

C. Performance in Course Assessments

Final projects and lab-based assessments were evaluated using a rubric-based approach, the criteria like accuracy, innovation, teamwork, presentation, and documentation. Fig.8 depicts the distribution of project scores across students.

1. Average project score: 74.2%
2. High-performing group (top 10 students): consistently scored above 85%, showcasing innovation in modular design and optimization.
3. At-risk group (bottom 10 students): scored between 55–60%, requiring additional mentoring in debugging and coding practices.

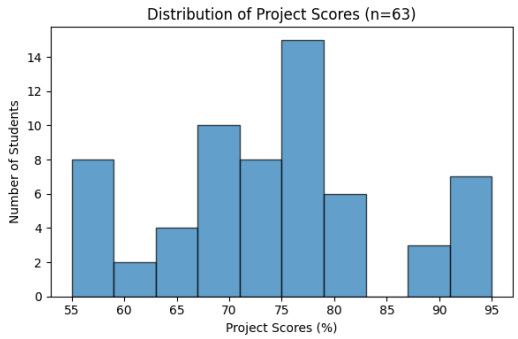

Fig.8. The distribution of project scores across 63 students.

D. Engagement Levels

Engagement was tracked via peer reviews, group discussions, and lab report submissions. The engagement breakdown of students is shown in Fig.9.

1. Active participation rate: ~82% (52 out of 63 students regularly engaged in peer/group tasks).
2. Report submission compliance: 95% of students submitted lab reports on time.
3. Group collaboration: Most groups distributed tasks equitably, though a few required instructor intervention.

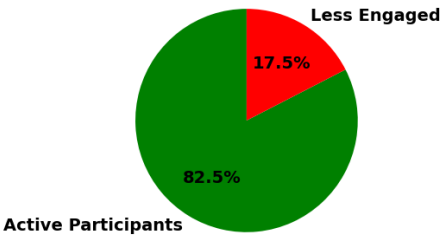

Fig. 9. Student Engagement Levels.

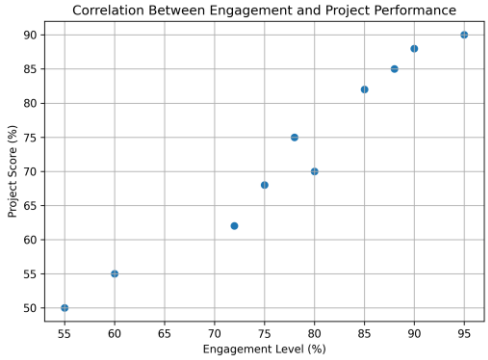

Fig.10. Correlation between engagement levels and project performance

A correlation analysis as shown in Fig.10, was included to illustrate the relationship between engagement levels and project outcomes.

E. Qualitative Feedback

Insights were gathered from student reflections and instructor observations:

Student Reflections

1. IBL encouraged them to "think like designers" rather than merely memorize syntax.
2. Peer review helped expose diverse approaches.
3. Reported challenges included debugging timing errors and mastering test benches.

Instructor Observations

1. Students demonstrated stronger problem-solving skills, exploring multiple strategies.
2. The quality of project presentations was notably higher than in lecture-based cohorts.
3. Time management during labs was identified as an area for improvement.

A word cloud was generated from student reflections. Key terms such as innovation, teamwork, problem-solving, and engagement highlight positive perceptions of IBL, while debugging and timing errors represent recurring challenges. Fig.10 shows the word cloud generated from students' reflective responses.

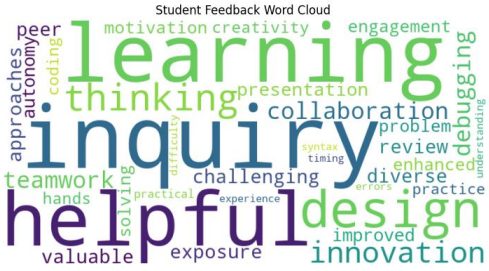

Fig.10. Student Feedback Word Cloud.

F. Summary of Findings

This subsection summarizes all quantitative and qualitative results of the analysis of the IBL framework. The findings reveal significant changes in the performance of the students, their involvement, and project output.

- Quantitative Outcomes: There was a ~25% improvement in student performance, quality of the project, and better engagement.

- Qualitative Insights: Students prioritized independence, teamwork, and design thinking, and the instructor's reported conceptualization and innovation.
- Overall Impact: The implementation of IBL in Digital System Design in Verilog greatly contributed to student learning performance that is in line with the suggested framework.

Table III provides the quantitative findings of the assessment, such as the pre-test and post-test averages, participation rates, project delivery, and statistical acceptability.

TABLE III
SUMMARY OF QUANTITATIVE FINDINGS FROM THE EVALUATION OF THE IBL FRAMEWORK (N = 63).

Metric	Value
Pre-test Average Score (%)	52.0
Post-test Average Score (%)	77.0
Learning Gain (%)	+25.0
Engagement (Active Participants)	82.5% (52 out of 63 students)
Project Score (Mean ± SD) (%)	74.6 ± 9.1
Statistical Significance (Paired t-test)	T (62) = 15.8, p < 0.001

DISCUSSION, CONCLUSION, AND FUTURE WORK

The results of this paper suggest that IBL is a viable method for Digital System Design using Verilog course and produces increased learning results for the students. The pre-test and the post-test comparison show clear improvement. Students gained both factual knowledge and deeper conceptual understanding. This is in line with previous research on STEM education, which reports that inquiry and active learning forms have continually achieved better performance by students over the traditional lecture-based education with lower failure rates.

The distribution of project scores also consolidates this finding. Although most students performed well, a significant number of them conceived very innovative designs, which indicates that IBL enables a fine balance between competence and creativity. Importantly, the analysis tracing the engagement on the level of discussion, collaboration, and peer review identified that most learners were actively engaged both in terms of engagement and the delivery of contributions. This is in line with findings of other literature about the possible roles of IBL in developing motivation in the learners, effective collaboration, as well as problem-solving capabilities (Y. A. Attia et al., 2019)

However, challenges were also identified. A subset of students had difficulty with syntax and debugging of complex constructs of Verilog. This implies that despite the advantages of exploration in IBL, novice learners might need to be scaffolded to avoid the risk of cognitive overload, as prior literature warns of such minimal guidance in inquiry-based models (P. A. Kirschner et al., 2006).

CONCLUSION

In general, the research proves that IBL is an effective learning method in DSD. It not only empowered students with technical skills but also developed the so-called transferable skills, including critical thinking, creativity, communication, and collaboration. These results demonstrate the greater possibilities of IBL to enhance contemporary engineering education and equip the graduates to solve real-life problems (A. Hmelo-Silver et al., 2004).

FUTURE WORK

Future work will involve extending the model to larger cohorts to evaluate scalability and robustness. Blended learning tools can also be incorporated, like the online Verilog simulators and FPGA-based virtual labs, to create accessibility and give real-time feedback. Longitudinal studies on students who were taken to higher levels of study (e.g., embedded systems, VLSI design) would also be interesting in evaluating long-range effects of IBL. Also, adaptive assessment techniques and AI-based feedback might be used to overcome challenges related to scaffolding, providing different learners with personalized feedback. The ultimate step to giving IBL credence as a tool of applicating in related engineering fields would serve as an additional measure that verifies its applicability across the board and establishes it as a tool of inculcating skills of innovation, independence, and problem solving among aspiring future engineers.

The work, therefore, focuses on empirical evidence that integrating IBL into Digital System Design using Verilog can simultaneously increase technical competence, creativity, and engagement. The approach establishes a bridge between theory and practice and trains the students not only as coders, but as problem-solvers ready to handle contemporary engineering problems.

APPENDIX

A. Sample Inquiry Task

The example inquiry task asked students to create a 4-bit synchronous counter in Verilog, using inquiry prompts that directed their work in both breaking down the problem and creating a testbench for that user-designed unit.

B. Anonymized Student Reflection

An example anonymous student reflection stated: "Now that I have experienced the effect of a testbench on my module, I have also learned to see issues such as timing clearly, as I was able to better understand what was happening while I was debugging."

C. Qualitative Coding Scheme

The qualitative coding framework summarized student work and included the following categories: Understanding the concept, the strategies used by students while debugging, Collaboration between team members, and the extent of inquiry among groups.

D. Pre/Post-Test Instruments

The Pre/Post-test tools used by the students consisted of 10 multiple-choice questions related to HDL syntax, interpretation of simulation output, and module design..

REFERENCES

Prince, M. (2004). Does active learning work? A review of the research. Journal of Engineering Education, 93(3), 223–231.

Dewey, J. (1938). Experience and education. Macmillan.

Creswell, J. W. (2012). Educational research: Planning, conducting, and evaluating quantitative and qualitative research (4th ed.). Pearson.

Chin, C., & Osborne, J. (2008). Students' questions: A potential resource for teaching and learning science. Studies in Science Education, 44(1), 1–39.

Felder, R. M., & Prince, M. (2006). The case for inductive teaching. In Proceedings of the ASEE Annual Conference.

Savery, J. (2006). Overview of problem-based learning: Definitions and distinctions. Interdisciplinary Journal of Problem-Based Learning, 1(1), 9–20.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark (2006). Educational Psychologist, 42(2), 99–107.

Alnajjar, H., Hassan, M. F., & Saad, A. H. (2018). Inquiry-based labs in Verilog learning: Enhancing student design skills. International Journal of Engineering Education, 34(3), 890–899.

Chang, Y., & Wang, H. (2018). Inquiry-based learning in digital logic design education. IEEE Transactions on Education, 61(4), 320–329.

Kumar, S., & Singh, P. (2019). Active inquiry pedagogy for hardware description language courses. International Journal of Computer Applications, 975, 8887–8893.

Chen, K., Liu, Z., & Xu, J. (2019). Inquiry-based FPGA projects in undergraduate engineering curriculum. Education and Information Technologies, 24(2), 1779–1793.

Yadav, A., Subedi, S., & Lundeberg, D. (2020). Inquiry-based approaches in computer science education: A systematic review. Computer Science Education, 30(1), 1–28.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences of the United States of America, 111(23), 8410–8415.

Strobel, J., & van Barneveld, A. (2009). When is PBL more effective? A meta-synthesis of meta-analyses comparing PBL to conventional classrooms. Interdisciplinary Journal of Problem-Based Learning, 3(1), 44–58.

Li, L., & Huang, Y. (2020). Simulation-based inquiry learning in Verilog design courses. IEEE Access, 8, 14563–14571.

Banerjee, R., Sharma, A., & Gupta, T. (2021). Inquiry-based Verilog laboratory practices to enhance digital system design. Education and Training in Digital Electronics, 12(2), 25–35.

Park, J., & Lee, C. (2021). Sequential circuit inquiry learning: Improving debugging skills in Verilog. IEEE Transactions on Learning Technologies, 14(3), 256–265.

Dasgupta, S., Nair, P. R., & Banik, M. (2021). Inquiry-driven project-based learning in VLSI design courses. International Journal of Engineering Pedagogy, 11(6), 45–56.

Rahman, A., & George, T. (2022). Inquiry-based learning strategies for Verilog-centric digital design education. Journal of Engineering Education Research, 28(2), 55–66.

Attia, Y. A., Ismail, M. H. H., & El-Khalili, R. A. (2019). Inquiry-based learning in computer engineering: Enhancing engagement and critical thinking. IEEE Transactions on Education, 62(4), 270–278.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. Educational Psychologist, 41(2), 75–86.

Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn? Educational Psychology Review, 16(3), 235–266.