Journal of Computer Technology and Applied Mathematics
Journal Home: http://jctam.suaspress.org | CODEN: JCTAAD
Vol. 3, No. 1, 2026 | ISSN 3007-4126 (Print) | ISSN 3007-4134 (Online) | ISSN 3007-4142 (Digital)

Human in the Loop AI EDA for Chip Design

ZHANG, Lei [1*]

[1] Clark University, USA

**** ZHANG, Lei is the corresponding author, E-mail: 18096833593@163.com***

Abstract: Currently, chip design automation is hampered by factors such as data scarcity, difficulty in sharing, and high costs associated with design rules and verification. Based on the research trajectory of human-machine collaborative learning, this paper proposes a modular framework that embeds expert feedback into three stages: data processing, model training, and system-level intervention, implementing it within the placement and routing workflow. With less human input, the number of iterations decreases. Economic accounting connects marginal quality gains with human and computing costs, exhibiting diminishing returns and context-dependent optimal feedback budgets.

Keywords: Human in The Loop Learning, Electronic Design Automation, Chip Placement and Routing, Preference Feedback, Cost–benefit Analysis.

Disciplines: Artificial Intelligence Technology. **Subjects:** Machine Learning.

DOI: https://doi.org/10.70393/6a6374616d.333732 **ARK:** https://n2t.net/ark:/40704/JCTAM.v3n1a05

1 INTRODUCTION

Chip design automation is increasingly surrounded by machine learning components, ranging from learned surrogate models that approximate expensive evaluations to decision policies that propose candidate actions in placement, routing, and iterative closure. This trajectory is understandable: the design space expands combinatorially, constraints are coupled across abstraction layers, and the opportunity cost of a locally suboptimal decision may propagate into late-stage rework, schedule slips, and uncertainty in downstream manufacturability and reliability. [1]At the same time, the prospect of fully autonomous pipelines remains, at least to some extent, fragile once one leaves curated benchmarks and confronts production realities, where rule decks evolve, tool versions change, and even the semantics of "valid actions" can be context-dependent, partially specified, and occasionally contested in engineering practice. Considering the above factors, it becomes difficult to treat chip design as a static optimization problem; it more closely resembles a moving target in which the target function, the feasible region, and the measurement apparatus all drift, sometimes slowly and sometimes abruptly.

1.1 WHY HUMAN-IN-THE-LOOP IS NOT MERELY A LABELING STRATEGY

A common, and perhaps overly convenient, reading of human-in-the-loop learning is that it mainly denotes manual labeling, and that its value is largely reducible to collecting more labels at lower cost. In chip design automation, labels are rarely atomic in the first place, since an engineer's intervention may manifest as a preference between two imperfect candidates, a constraint edit that encodes tacit manufacturability knowledge, or a sequence of corrective actions whose merit becomes visible only after several downstream tool stages. When we attempted to translate generic human-in-the-loop recipes into a design-flow setting, an early difficulty was definitional rather than algorithmic: what exactly constitutes a "key instance" in placement or routing, where a local hotspot may later be alleviated by timing-driven re-optimization, or may instead become the seed of cascading violations, and where the same symptom can be produced by distinct underlying causes. We also observed that naive uncertainty-based querying can over-select visually ambiguous yet economically irrelevant cases, while missing sparse, structurally critical situations in which a small human correction plausibly prevents many later iterations; this led us to treat key-instance discovery as a design problem in its own right, rather than as a detachable module imported from standard active learning. A further complication is that feedback is not guaranteed to be stable across annotators, not because experts are careless, but because they may optimize different implicit objectives under different time pressures, which suggests that the feedback channel itself should be modeled as informative but imperfect.[2]

1.2 ROBUSTNESS, DISTRIBUTION SHIFT, AND THE NECESSITY OF SYSTEM-LEVEL INTERACTION POLICIES

Human-in-the-loop systems are often expected to handle domain changes, disturbances, and out-of-range

Journal of Computer Technology and Applied Mathematics
Journal Home: http://jctam.suaspress.org | CODEN: JCTAAD
Vol. 3, No. 1, 2026 | ISSN 3007-4126 (Print) | ISSN 3007-4134 (Online) | ISSN 3007-4142 (Digital)

samples with a level of robustness that purely offline models struggle to achieve, yet chip design pipelines embody precisely the kinds of distributional instability that make robustness both crucial and elusive. Designs differ in scale, hierarchy, and constraint regimes; industrial contexts differ in risk tolerance and closure priorities; tool heuristics differ across versions; and engineering teams differ in how they operationalize "good enough" under schedule constraints. Under these conditions, it is plausible that human input is most valuable not when the system is operating within its comfort zone, but when it detects that it is outside it, which naturally pushes the framework toward system-level interaction policies that decide when to query, what to query, and how to incorporate feedback without destabilizing the flow. This leads us to further thinking about interaction as a controlled resource allocation problem rather than a fixed annotation budget, where the system must negotiate the trade-off between being overly conservative and excessively autonomous.[3]

1.3 From Technical Improvement to Economic Meaning

Chip design decisions are not only technical; they are also economic, even when the immediate objective is framed as power, performance, and area. Human time is scarce, compute resources are priced, and iteration counts translate into calendar time under organizational constraints, which means that a learning system that improves a metric but consumes disproportionate expert attention may not be practically attractive. [4]During early prototyping, we encountered an instructive asymmetry: small doses of expert feedback sometimes reduced constraint violations and shortened certain closure loops, yet the same feedback did not consistently translate into proportional gains in final power, performance, and area, and multiple explanations remained plausible. The feedback may have been rationally targeted at feasibility rather than global optimality; the benchmark distribution may have favored designs where constraint fixes have high leverage; engineers may adapt to the presence of an assistant by reallocating their own effort, effectively shifting the baseline in ways that complicate causal attribution. These possibilities do not invalidate human-in-the-loop approaches, but they suggest that evidence should be interpreted with caution, that alternative mechanisms should be considered, and that further research is needed under workloads that better resemble industrial operating conditions.

2 Related Work and Conceptual Foundations

The recent wave of "AI for chips" is often narrated as a design automation story, yet in practice it is also a manufacturing and economic story, because the quality of upstream data, the stability of production operations, and the measurability of downstream value jointly decide whether an AI intervention is merely demonstrable or genuinely deployable. [5]A useful starting point is the observation that many chip production environments are information-rich but decision-poor sensor streams are dense, but labels for rare faults, root-cause attribution, and counterfactual outcomes remain scarce. This asymmetry tends to push researchers toward proxies, synthetic augmentation, or narrow task definitions, which can be productive while still leaving open questions about external validity and long-horizon impact on yield, cycle time, and cost.[6]

A recurring bottleneck is not model capacity but data readiness. Work on automated extraction–transformation–load pipelines and imbalance-aware preprocessing treats data quality control as an operational system rather than a one-off cleaning step, emphasizing that missingness patterns, noise, and skewed defect classes can jointly bias predictive signals even before a learner is selected. [7]The methodological choice is important here: when "better performance" is obtained primarily by interpolation rules, sampling strategies, and feature voting, the resulting gains may be robust in the short run while still depending on the stability of the data generating process, which is not guaranteed in fabs where tool drift, recipe changes, and maintenance cycles shift distributions. This suggests that data engineering and model choice are entangled, and it becomes difficult to attribute improvements to the algorithm alone, a difficulty that later motivates this paper's emphasis on evaluation designs that separate data pipeline effects from learning effects.

Predictive maintenance provides a closely related lens, because its value is simultaneously technical and economic. A stacking-based classifier combined with explainability tools such as SHAP and LIME illustrates a pragmatic attempt to bridge accuracy and actionability, while synthetic data is used to compensate for limited fault incidence [8]. This combination is appealing, yet it also raises questions that matter for deployment: synthetic augmentation can reduce variance and help training, but it may imprint the assumptions of the simulator into the decision boundary; explainability may improve human trust, but it can also create a false sense of causal certainty if explanations are computed on correlational features under dataset shift. Considering these tensions, predictive maintenance studies are best read not simply as "high accuracy achieved," but as prototypes of a larger human decision loop in which maintenance teams, tool engineers, and planners interpret model outputs under real constraints.[9]

Beyond maintenance, semiconductor manufacturing research increasingly frames performance degradation as a dynamic bottleneck phenomenon. A bottleneck detection approach that combines an active-period style method with visualization makes the diagnosis legible to operators, and it implicitly acknowledges that bottlenecks are not purely statistical objects but managerial objects that must be communicated and acted upon [10]. The method's strength lies in its operational interpretability, yet it also invites further scrutiny on measurement granularity: if bottlenecks are

scored using uptime cycles, the score may conflate true capacity constraints with scheduling policies, preventive maintenance windows, or product-mix variability. In that sense, bottleneck analytics sits at the interface between descriptive monitoring and prescriptive intervention, and it is precisely where economic objectives, such as throughput maximization under WIP constraints, begin to interact with learning objectives.

Defect prediction under small and imbalanced datasets pushes this interface further. An explainable AutoML framework that searches model classes and hyperparameters while explicitly handling imbalance aims to deliver both performance and "actionable" feature insights.[11] The attractiveness of AutoML is that it can reduce manual tuning burden, but its limitations are also relevant: search procedures may overfit to a benchmark split, interpretability modules may inherit instability when features are correlated, and the automated choice of loss functions or sampling schemes may encode implicit trade-offs that practitioners would prefer to control explicitly. From the perspective of this paper, these works motivate a more explicit articulation of objective functions and constraints, because interpretability alone does not settle which errors are economically tolerable.

Operational constraints become most visible in wafer logistics and scheduling, where uncertainty is structural rather than incidental. A robust bilevel network-flow scheduling formulation under WLTP uncertainty illustrates an attempt to formalize hierarchical decision-making, in which upper-level policies and lower-level flows interact. [12]The bilevel structure is conceptually aligned with real fabs, where dispatching rules, tool dedication, and transport constraints cascade across levels. At the same time, bilevel optimization is computationally demanding, and robustness often depends on how uncertainty sets are defined; if those sets are misspecified, "robust" solutions can be conservative in the wrong direction. This reinforces a broader point: in semiconductor operations, optimization and learning are rarely separable, since learning informs forecasts and uncertainty models, while optimization changes the data that will be observed next.

Human–machine collaboration enters the picture not as a slogan, but as a necessity when systems become too complex to supervise manually. An IoT-enabled ambient sensing system coupled with large language models for complex activity tracking highlights how modern pipelines increasingly rely on multi-stage perception and language-based reasoning to summarize events and coordinate actions. [13]While the application domain is not limited to semiconductor manufacturing, the architectural lesson is transferable: when multiple components observe, interpret, and report system state, errors can compound across stages, and the human role shifts from direct control to supervision, auditing, and exception handling. This raises practical questions about accountability and calibration: when an LLM-generated narrative suggests a causal chain, operators may treat it as explanation rather than hypothesis, so governance mechanisms and uncertainty communication become part of system design rather than post-hoc documentation.[14]

A technical barrier to such collaboration is that real industrial data is frequently incomplete and multi-modal, with block-missing patterns that cannot be treated as random dropout. Multi-response regression methods for block-missing multimodal data without imputation emphasize modeling strategies that avoid naive filling-in, which can otherwise inject artifacts and distort uncertainty. [15]Complementary dissertation work on supervised learning for complex data provides a broader methodological backdrop for handling high-dimensional, structured, and partially observed inputs in a principled way. [16]These contributions are relevant because chip design and manufacturing analytics increasingly combine heterogeneous signals, including layout-derived features, tool sensor streams, and textual maintenance logs; if missingness is systematic, both model selection and evaluation can be biased unless the missingness mechanism is explicitly considered.

The economic layer is not merely an "application context," because it supplies the criteria by which success is judged. Incentive mechanisms for teamwork based on unfairness aversion preferences remind us that organizational behavior shapes how tools are adopted, how data is shared, and how model recommendations are followed. [17]In chip R&D organizations, human–AI systems can fail not because predictions are inaccurate, but because incentives discourage disclosure of anomalies, penalize short-term throughput dips needed for long-term yield improvement, or concentrate decision rights in a way that reduces feedback quality. This suggests that economic modeling should be embedded early, with explicit attention to incentive compatibility, not appended as a discussion after technical results.

Finance-oriented AI studies, although far from the fab floor, can still inform the economic measurement problem because they treat constraints and generative uncertainty as first-class modeling objects. Deep learning asset pricing under no-arbitrage constraints illustrates an approach in which domain constraints are integrated into learning, rather than enforced after training, which parallels how physical and operational constraints might be integrated into chip design optimization pipelines. [18]Generative diffusion models for option pricing, which aim to model volatility dynamics and tail behavior, similarly foreground uncertainty generation rather than point prediction, and this is conceptually relevant when the outcomes of interest are distributions of cycle time, yield, or cost under rare disruptions. [19]The analogy should not be overstated, but it does suggest that constraint-aware learning and generative uncertainty modeling can enrich the evaluation of chip-related AI systems, especially when decision-makers care about worst-case or tail-risk outcomes rather than average metrics.[20]

Taken together, the literature implies a set of interlocking bottlenecks that this paper treats as a unified

Journal of Computer Technology and Applied Mathematics
Journal Home: http://jctam.suaspress.org | CODEN: JCTAAD
Vol. 3, No. 1, 2026 | ISSN 3007-4126 (Print) | ISSN 3007-4134 (Online) | ISSN 3007-4142 (Digital)

research problem. Data quality and missingness limit what models can learn; interpretability and human supervision shape what recommendations can be acted upon; optimization and scheduling constraints shape what outcomes can be realized; incentives and value measurement shape what is adopted and sustained. This motivates a research design that does not treat "chip + AI + economics" as three separable topics, but as a coupled system in which technical performance must be translated into operational decisions and then into economically meaningful outcomes, acknowledging that each translation stage can introduce its own failure modes and biases.[21]

Motivated by these practical frictions, this paper asks how human input should be positioned along the chip design automation pipeline so that it improves outcomes under distribution shift rather than only on a static benchmark, how expert feedback can be represented so that it captures higher-dimensional engineering knowledge beyond incremental label accumulation, and how the resulting technical gains relate to marginal labor and compute costs so that an approximate, context-dependent operating point can be identified rather than assumed. In pursuing these questions, this paper develops a modular human-in-the-loop framework that links key-instance selection, preference- and constraint-informed learning signals, and interaction triggers that are sensitive to risk and drift; it evaluates the framework in a placement and routing workflow using reproducible settings while documenting the non-idealities that arise from tool instability and feedback heterogeneity; and it complements technical metrics with an explicit cost–benefit accounting that reveals plausible diminishing returns and highlights where the approach may or may not generalize, thereby positioning the contribution as a structured empirical exploration rather than a claim of universal superiority.[22]

3 HUMAN IN THE LOOP FRAMEWORK FOR AI EDA

3.1 DESIGN OBJECTIVES, SCOPE, AND WHAT COUNTS AS AN INTERVENTION

This section presents a human-in-the-loop framework for learning-assisted EDA that is intended to be implementable in a placement and routing workflow, while remaining explicit about the boundaries of what the system can plausibly guarantee. The framework is designed around a practical observation: in chip design, the most consequential errors are often not the most frequent ones, and the most frequent signals are not always the most actionable. [23]A model can look competent on average, yet still fail in the rare but costly situations that trigger long closure cycles or late-stage engineering change orders. Considering the above factors, the core goal is not to maximize a single metric at any cost, but to reduce the number of expensive iterations and high-risk failures under a fixed budget of human attention and compute time, while keeping the final quality within an acceptable band.

A central choice is the definition of the "unit of intervention." In textbook supervised learning, the unit is usually an example paired with a label.[24] In AI-EDA, the objects that matter are heterogeneous: a placement state with emerging congestion, a routing attempt that creates a design-rule dead end, a timing closure configuration that makes the optimizer oscillate, or a constraint template that quietly biases the search toward brittle solutions. Treating all of these as identical "samples" can oversimplify how designers actually think and act. The framework therefore treats human input as structured feedback attached to decision points, rather than as a single label type, and it tries to preserve enough context that the same feedback can be reinterpreted later if the objective weights or constraint priorities change.[25]

We organize the framework around three coupled objectives. The technical objective is to improve feasibility and convergence, with performance and area treated as joint but not perfectly aligned targets. The robustness objective is to keep the system usable when design distributions shift across blocks, when constraint regimes evolve within a project, and when tool settings are retuned. The economic objective is to translate improvements into marginal gains per unit of scarce expert time and per unit of compute, since practical adoption often depends less on whether an approach can win a benchmark and more on whether it can lower the total cost of reaching an acceptable solution.[26]

3.2 PIPELINE OVERVIEW AND INTERFACES TO THE EDA FLOW

The framework assumes an EDA toolchain that can be executed iteratively on a design instance. We treat the toolchain as an oracle that, given a design state and an action proposal, returns observable outcomes, such as congestion metrics, estimated timing indicators, violation counts, and runtime. This black-box stance is partly forced by practical constraints, since differentiating through industrial tools is rarely feasible, and partly chosen to improve portability across tool stacks. Within this setting, the system exposes four interfaces that define how learning, human input, and the EDA engine interact.[27]

State encoder: The encoder maps a design state into a structured representation. In practice, we use a graph representation derived from connectivity, spatial features derived from placement and congestion maps, and constraint descriptors that indicate which objectives are active and which rule classes are binding. The encoder is designed to be stage-aware, since the relevance of features differs between early placement exploration and late routing repair.

Candidate generator: The generator proposes a small set of candidate actions or candidate partial solutions. In placement, candidates may be alternative macro placements, alternative legalization moves, or alternative local perturbations around a congested region. In routing,

candidates may be alternative priority settings, region constraints, or localized repair tactics. We keep the generator modular because the "correct" action space is rarely stable across tool versions and design styles, and the system should degrade gracefully if a subset of candidates becomes invalid under a new rule deck.[28]

Interaction controller: The controller decides when to query a human, what to display, and how to record the response. It is designed to be auditable, meaning that each query has a logged rationale and each response has a logged transformation into a learning signal. This is not only for debugging; it is also for governance, because in high-stakes design contexts, a system that cannot explain why it requested intervention is difficult to trust.

Learner and optimizer. The learner updates internal models from tool outcomes and human feedback. The optimizer uses these models to rank candidates, allocate exploration effort, and decide whether to request further feedback. The learner is not a single model by necessity. It can include a score model for ranking, a calibration module for uncertainty, and a policy module that proposes candidate sets. This modular pipeline is intentionally not "end-to-end" in a strict sense. In our early prototypes, end-to-end ambitions repeatedly collided with interface fragility. A policy that proposes fine-grained actions can be invalidated by small tool changes, and an end-to-end objective can become ambiguous when constraints are updated midstream. A modular decomposition is not aesthetically pure, yet it is often operationally more stable.

3.3 KEY-INSTANCE IDENTIFICATION AS A DECISION PROBLEM UNDER SCARCITY

Human attention is scarce, and not all intervention opportunities are equally valuable. The framework treats key-instance identification as a decision problem under partial information: at each iteration, the system observes a pool of candidate states and actions, then selects a small subset for human review that is expected to maximize downstream benefit under a time budget.

A simplistic strategy is to query when the model is uncertain. In chip design, uncertainty is not necessarily aligned with impact. Some ambiguous-looking states resolve naturally through later optimization, while some seemingly ordinary states trigger expensive cascades because they interact with hidden constraints or future routing feasibility. We found that a practical controller needs to consider multiple signals, and also needs to learn from its own mistakes about what was worth asking.

We implement key-instance scoring as a combination of three signals. A proxy that the current trajectory is heading toward hard failures, such as rapidly increasing violation counts, repeated oscillations in closure metrics, or signs that routing resources are saturating. This signal is noisy early in the flow, since early estimates are coarse, yet it becomes informative as the design approaches closure. Leverage potential. A proxy that a small correction could substantially change the downstream trajectory. We estimate leverage by comparing predicted score distributions across the candidate set and flagging situations with high spread, since high spread suggests that choice among candidates matters. This proxy is imperfect. High spread can also reflect model instability rather than true leverage. Still, it provides a principled way to distinguish "many candidates are similarly good" from "a bad choice will be expensive."

Distributional novelty: A proxy that the system is operating outside its recent experience. We estimate novelty by representation distance in the encoder space and by the frequency of rule patterns observed in the logs. Novelty is a double-edged signal. It can catch genuine drift, but it can also over-trigger on benign differences. We therefore treat novelty as a modulator that increases caution, rather than as a sufficient condition to query. During development, these signals often disagreed. Some novelty triggers produced no downstream benefit, suggesting the controller was being overly conservative. Some leverage triggers appeared late, after costs had already accumulated, implying we were reacting rather than anticipating. We adjusted by letting the controller update its internal weights based on realized payoffs, and by introducing a memory that penalizes query patterns that repeatedly produce low benefit. This is not a guarantee of optimal behavior; it is an attempt to make the controller less naive and more economically aligned.

3.4 HUMAN FEEDBACK PROTOCOL AND REPRESENTATION

The system supports three feedback types, selected to balance cognitive load, information content, and auditability. Comparative preference feedback. The expert is shown two or more candidate states or candidate solutions and is asked to rank them or select a preferred option. Preference feedback aligns with how designers often reason, by comparing trade-offs rather than declaring a single correct label. It also avoids forcing absolute judgments on structured outputs. However, preferences can be inconsistent across experts, especially when candidates trade congestion against timing in subtle ways. Rather than treating inconsistency as noise to be suppressed, we log annotator confidence and rationale notes, since disagreement can reflect genuine multi-objective ambiguity.

The expert can apply a small constraint change or localized edit, such as restricting a macro region within permissible bounds, adding a keepout, adjusting a routing guide, or prioritizing a constraint class for a repair. Constraint edits encode high-level feasibility knowledge, yet they also modify the environment rather than merely labeling it. This creates a reproducibility risk. A constraint edit must be captured in a replayable format, or else subsequent runs may not reflect the same intervention, making evaluation fragile. We treat each edit as a versioned artifact with provenance,

Journal of Computer Technology and Applied Mathematics
Journal Home: http://jctam.suaspress.org | CODEN: JCTAAD
Vol. 3, No. 1, 2026 | ISSN 3007-4126 (Print) | ISSN 3007-4134 (Online) | ISSN 3007-4142 (Digital)

and we store the pre-edit and post-edit states.

For selected high-impact failures, the expert performs a short sequence of repair actions, and the system records intermediate states. Repair traces are information-rich, but they are expensive. We reserve them for cases where the controller predicts that a failure will be repeated, and where preference feedback alone is unlikely to reveal how experts navigate feasibility constraints. All feedback events are stored as tuples containing the state representation, the candidate set, the chosen preference or edit, the human time spent, and the downstream outcomes after applying the chosen action and rerunning the toolchain. This unified representation is important for two reasons. First, it supports multiple learning objectives without forcing us to reformat data at every stage. Second, it allows retrospective analysis when we later discover that an objective weight, a rule deck, or a tool version changed, which is more common than one might want to admit. The score model predicts a scalar utility for each candidate, intended to correlate with downstream feasibility and quality. The score is constructed from observable metrics, such as violation counts, timing proxies, congestion measures, and runtime, with weights that reflect stage priorities. We treat the weight choice as an explicit modeling decision rather than a hidden constant, because different teams implicitly optimize different utility functions under different schedule pressures. Even within a team, priorities can change mid-project, which implies that any learned scoring model may require recalibration. Tool outcomes provide noisy but plentiful signals. They reflect what the toolchain measures, which is not always what the team ultimately values, yet they are available at scale.

Human feedback provides sparse but structured signals. Preference data is used through a ranking loss that encourages the model to assign higher scores to preferred candidates. Constraint edits and repair traces are used as demonstrations, with the model learning to assign higher utility to states that resemble post-edit or post-repair conditions, subject to feasibility constraints. The hybrid training objective attempts to reduce proxy misalignment. Still, it is not guaranteed to fully resolve misalignment, because human preferences can also be context-dependent. A designer might prefer a conservative fix under schedule pressure and a more aggressive fix early in the project, even if the local metrics are similar. This suggests that preference models may need to be conditioned on project phase or on risk tolerance indicators, a direction that is plausible yet not fully settled in current evidence.

Policy model: The policy model proposes candidates. In early trials, we experimented with a more direct policy that outputs a single action, but we repeatedly encountered feasibility failures when constraints changed or when tool settings were retuned. The system became brittle in a way that was hard to debug. We shifted to a conservative policy that proposes a candidate set, leaving final selection to the optimizer and, when queried, to the human. This sacrifices some autonomy, but it improves safety and interpretability, and it makes the human role more naturally supervisory rather than corrective.

3.5 Interaction Controller: Escalation Logic, Calibration Checks, and Drift-Aware Behavior

The interaction controller orchestrates the human and machine roles. It decides whether to ask a quick preference question, to request a constraint edit, or to escalate to a repair trace. The controller follows a tiered strategy: it prefers low-cost feedback and escalates only when evidence suggests that low-cost feedback is unlikely to change outcomes. Embedding shift monitoring. The controller tracks distribution statistics of state embeddings over time. If the embedding distribution shifts beyond a threshold, the controller increases query frequency and reduces reliance on model confidence. This is not because more feedback is always beneficial, but because confidence becomes less informative when the system is out of distribution. Outcome calibration monitoring. The controller compares predicted scores against realized outcomes. Persistent miscalibration triggers additional queries and an increase in exploration to rebuild a reliable score model. Calibration monitoring is imperfect. Outcome noise can arise from tool nondeterminism, from fluctuating runtime environments, or from measurement error in proxies. Still, it gives a practical signal that the system's internal assumptions may have become stale.

The framework is designed to be reproducible, yet several practical frictions repeatedly complicated our own implementation, and it is more honest to record these frictions than to present an overly clean pipeline. Tool interfaces were not always stable across environments. Minor configuration differences sometimes produced different intermediate artifacts, which changed the measured outcomes. We responded by capturing configuration files, seeding what could be seeded, and defining evaluation statistics that are robust to small perturbations, focusing on distributions and medians rather than single runs. Even with these precautions, full determinism remained difficult, suggesting that evaluation should be framed probabilistically rather than as a single number. Feedback heterogeneity also mattered. Experts disagreed, not only due to noise, but because they implicitly optimized different objectives. We initially attempted to enforce strict consensus, but it slowed progress and did not reliably improve learning. We shifted to a representation that preserves disagreement and uses confidence weighting, treating feedback as informative but imperfect. This raises open questions. Are there principled ways to infer latent utility functions of different annotators? Should the system adapt to a specific team's preferences rather than average across them? These questions remain plausible directions for further research.

Journal of Computer Technology and Applied Mathematics
Journal Home: http://jctam.suaspress.org | CODEN: JCTAAD
Vol. 3, No. 1, 2026 | ISSN 3007-4126 (Print) | ISSN 3007-4134 (Online) | ISSN 3007-4142 (Digital)

4 EXPERIMENTAL SETUP AND EVALUATION PROTOCOLS

4.1 TASKS, INSTANCES, AND WHAT WE TREAT AS A COMPARABLE "TRIAL"

The experimental setting is organized around a placement and routing workflow, not because these stages exhaust the design problem, but because they expose a dense interaction between feasibility constraints, proxy objectives, and iterative closure behavior, which is precisely where a human-in-the-loop policy is likely to face both its opportunity and its fragility. Each trial is defined as an end-to-end run over a design instance under a fixed constraint template and a fixed tool configuration snapshot, with intermediate states and decisions logged at pre-specified checkpoints. We deliberately avoid defining a trial as a single step decision, since such a definition can make improvements look sharper than they are by ignoring delayed effects, and it can also hide the fact that what appears locally optimal may lead to later dead ends.

Design instances are drawn from publicly available benchmarks that contain a mixture of macro-heavy and standard-cell-dominant blocks, and we stratify the evaluation so that a reported aggregate does not silently overrepresent the instances that are easiest to improve. In early exploratory runs, we found that macro-heavy blocks were prone to visually salient congestion patterns that encouraged frequent human queries, while some standard-cell cases produced fewer obvious triggers yet still accumulated subtle timing-related regressions, which suggested that instance composition could bias conclusions unless explicitly controlled. This observation motivated a stratified reporting protocol in which each instance is reported individually before any pooled summary is constructed.

4.2 TOOLCHAIN CONFIGURATION AND EXECUTION ENVIRONMENT

The EDA flow is executed through an open, scriptable placement and routing pipeline that supports repeated runs and state extraction, including congestion proxies, timing estimates, and violation statistics. Tool settings are captured as versioned configuration artifacts, and the configuration is treated as part of the experimental condition rather than as background detail. This emphasis was not purely methodological. We encountered small but non-negligible differences across machines and across runtime environments, where identical scripts produced slightly different intermediate results, particularly in late routing repair where heuristic tie-breaking can vary with runtime order. Instead of assuming full determinism, we designed the protocol to accommodate mild nondeterminism through repeated runs and distributional summaries, and we interpret single-run improvements as suggestive rather than definitive when variance is visible.

To reduce confounding, each condition is executed under a fixed compute budget. The budget includes tool runtime limits and model inference limits, since a method that "wins" by silently consuming substantially more runtime may not be comparable in a practical sense. We also log wall-clock time, because tool time and model time interact in non-additive ways when parallelization and caching are present.

4.3 HUMAN PARTICIPANTS, TRAINING, AND FEEDBACK ELICITATION

Human feedback is collected from participants with prior exposure to chip design concepts, including placement, congestion, and constraint reasoning. Before formal trials, participants complete a calibration session in which they review a small set of pilot instances and receive a standardized explanation of what the system will display and what the requested feedback means. We adopted this step after noticing that early feedback contained avoidable variability arising from interface misunderstanding rather than genuine preference differences. Even after calibration, disagreements persisted in multi-objective cases, and we chose not to force consensus because it can artificially compress the range of valid engineering judgments.

The interaction protocol supports three feedback modalities, preference selection among candidates, localized constraint edits, and short repair traces. Preference selection is used most frequently because it tends to impose a lower cognitive burden, yet we do not assume that low burden implies low value. In several pilot rounds, preference feedback corrected the system's tendency to overprioritize easy feasibility fixes at the cost of longer-term closure. Constraint edits and repair traces are reserved for cases that the interaction controller classifies as high risk or high leverage, partly because they are costlier in time, and partly because they can change the environment in ways that complicate replay. For every interaction, we log elapsed time, self-reported confidence, and a short textual rationale, not because rationale is always reliable, but because it provides a useful lens for later error analysis when the same technical outcome can be produced by different reasoning paths.

A practical concern is fatigue and attention drift. Human-in-the-loop experiments can inadvertently treat human input as a stationary resource, while in reality attention degrades with repetition, and experts adapt to the system's behavior. We address this by randomizing the order of conditions, limiting session length, and inserting repeat questions that are designed to detect within-session consistency changes. We evaluate against three families of baselines. A fully automated baseline runs the same toolchain without human queries, using either default heuristics or a fixed learned ranking model that does not receive human feedback. This baseline captures the improvement that comes from automation alone.

A human-guided baseline uses human decisions at the same checkpoints but without learned assistance, meaning

that participants choose among candidates or apply repairs without model-provided ranking. This baseline is imperfect, since humans are not optimized for exhaustive search, yet it provides a practical reference for whether the system is genuinely reducing cognitive effort or merely shifting it. The proposed human-in-the-loop condition uses the full framework, including key-instance selection, the tiered query policy, and the hybrid learning objective that integrates tool outcomes with human preferences and edits. To probe sensitivity rather than report a single headline number, we also evaluate different feedback budgets, ranging from sparse querying to more frequent querying, since the marginal returns of additional human time are a central object of interest rather than an incidental detail.

At the design-quality level, we track proxy measures of power, performance, and area where available, along with wirelength and congestion statistics, and we report constraint violations, including routing and rule checks, as direct feasibility indicators. These are standard EDA outputs, yet their interpretation is not always straightforward. A reduction in violation counts can reflect genuine improvement, but it can also reflect a change in constraint tightness or an unintended shift in routing strategy. For this reason, we report both absolute values and changes relative to the baseline under the same constraint template. At the process level, we track iteration count to reach a stable closure criterion, as well as the number of late-stage repair cycles. We treat these as economically meaningful because they proxy schedule risk and engineering rework, although the mapping from iteration count to calendar time is not constant across organizations. At the resource level, we report human time and compute time. Human time is measured in total interaction time and in the number of high-effort interventions, such as repairs, since a system that replaces many small questions with a few long repairs may have different operational implications. To avoid overinterpreting small differences, we require that an improvement be stable across repeated runs or across participant subsets, at least to some extent, before describing it as systematic. When improvements appear only in a subset of instances, we report them as instance-conditional, and we treat this heterogeneity as a result rather than an inconvenience.

4.4 Experimental Design, Randomization, and Ablation Logic

The evaluation combines two complementary tracks. A live-interaction track measures the system under real-time human feedback, capturing the practical interplay among query timing, human attention, and model updates. This track is closest to the intended use case, yet it is also the hardest to control. Human learning effects can contaminate comparisons, and small interface frictions can dominate results. We mitigate this through within-subject randomization, balanced condition ordering, and repeated calibration questions.

A replay track uses recorded human feedback and recorded tool outcomes to simulate interaction decisions under controlled conditions. Replay cannot fully capture real-time cognition, yet it allows more extensive ablations, including variants of the key-instance selector and the escalation policy. Replay also helped us diagnose a failure mode that was difficult to see in live runs, namely that a novelty trigger can increase query frequency in a way that crowds out high-leverage cases under a fixed budget, which is a subtle form of opportunity cost. Ablations are structured to isolate which part of the framework is carrying signal. We disable key-instance selection while keeping learning active, disable preference learning while keeping key-instance selection active, replace drift-aware triggers with fixed thresholds, and vary the candidate set quality while holding the interaction budget fixed. This pattern is chosen because interaction effects are not additive in a simple way, and a single ablation rarely explains behavior unless it is paired with a budget constraint that makes trade-offs explicit.

Robustness is evaluated through controlled shifts. We vary constraint templates within a bounded range, change design instance families, and perturb tool settings in ways that mimic reasonable retuning rather than adversarial sabotage. We also examine out-of-range behavior by exposing the system to instances whose structural statistics differ from the training pool, acknowledging that such a test is only a proxy for the more complex drift seen in real projects. When outcomes improve, we explicitly examine alternative mechanisms. An observed reduction in iteration count could arise from better candidate ranking, from humans being nudged toward more conservative fixes, or from the interaction policy suppressing risky exploration. Each mechanism has different implications for long-term quality and for economic value. Likewise, if PPA proxies improve, it may reflect genuine optimization, but it may also reflect benchmark composition or the fact that humans, once assisted, reallocate their effort to the remaining hard cases. These interpretations are not mutually exclusive, and the protocol is designed to reveal which ones are plausible in a given setting rather than forcing a single narrative.

4.5 Reproducibility, Logging, and Ethical Handling of Expert Labor

All trials produce structured logs that include tool configurations, intermediate states, candidate sets, model scores, query decisions, human responses, and downstream outcomes. This logging is essential for reproducibility, and it is also essential for accountability, since a human-in-the-loop system that cannot be audited is difficult to justify in high-stakes design contexts. We also treat expert labor as a resource that deserves ethical consideration. Participants are informed about task scope, session length is bounded, and the protocol avoids excessive repetition that yields diminishing research value while increasing fatigue. Although these steps do not eliminate all biases, they reduce the risk that the evaluation is shaped more by human exhaustion than by the

technical properties of the framework, and they support a more credible interpretation of cost–benefit trade-offs in later analysis. Considering the above factors, the evaluation protocol is designed less as a single scoreboard and more as an instrument for discovering where the framework is stable, where it is brittle, and how its benefits scale with the scarce inputs it consumes. This leads naturally to the next section, where empirical results are reported alongside robustness evidence and alternative explanations, rather than as a single, overly smooth performance narrative.

5 CONCLUSION

This paper explores a human-machine collaborative AI-assisted layout and routing paradigm that treats expert opinions as a scarce and economically meaningful input, rather than a free source of labels. The proposed framework links learning signals with constraint modifications and preference structures, while making costs visible through marginal accounting. Furthermore, it demonstrates that human-machine collaborative AI exploratory data analysis is not only a method to improve objective scoring, but also a design principle for building auditable and reliable optimization systems.

ACKNOWLEDGMENTS

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

FUNDING

Not applicable.

INSTITUTIONAL REVIEW BOARD STATEMENT

Not applicable.

INFORMED CONSENT STATEMENT

Not applicable.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

CONFLICT OF INTEREST

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

PUBLISHER'S NOTE

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

AUTHOR CONTRIBUTIONS

Not applicable.

ABOUT THE AUTHORS

ZHANG, Lei

Clark University, USA.

REFERENCES

[1] Chowdhury, A. B. (2024). Data-Driven EDA: Harnessing the Power of Machine Learning for Chip Design (Doctoral dissertation, New York University Tandon School of Engineering).

[2] Amuru, D., & Abbas, Z. (2024). AI-Assisted Circuit Design and Modeling. In AI-Enabled Electronic Circuit and System Design: From Ideation to Utilization (pp. 1-40). Cham: Springer Nature Switzerland.

[3] Chen, Y. (2025). Artificial Intelligence in Economic Applications: Stock Trading, Market Analysis, and Risk Management. Journal of Economic Theory and Business Management, 2(5), 7-14.

[4] Huang, S. (2025). Prophet with Exogenous Variables for Procurement Demand Prediction under Market Volatility. Journal of Computer Technology and Applied Mathematics, 2(6), 15-20.

[5] Cao, S., Wang, J., & Tse, T. K. (2023). Life‐cycle cost analysis and life‐cycle assessment of the second‐generation benchmark building subject to typhoon wind loads in Hong Kong. The Structural Design of Tall and Special Buildings, 32(11-12), e2014.

[6] Yin, M. (2025). Predictive Maintenance of Semiconductor Equipment Using Stacking Classifiers and Explainable AI: A Synthetic Data Approach for Fault Detection and Severity Classification. Journal of Industrial Engineering and Applied Science, 3(6), 36-46.

[7] Kanagal, R. (2025). LLM-Powered EDA Log Analysis for Effective Design Debugging.

[8] Wang, J., Kudagama, B. J., Perera, U. S., Li, S., & Zhang, X. (2025). Framework for generating high-resolution

Hong Kong local climate projections to support building energy simulations. Physics of Fluids, 37(3).

[9] Wang, J., Golnary, F., Li, S., Weerasuriya, A. U., & Tse, K. T. (2024). A review on power control of wind turbines with the perspective of dynamic load mitigation. Ocean Engineering, 311, 118806.

[10] Wang, C., Yang, F., & Zhu, K. (2024, August). AI-Enabled Layout Automation for Analog and RF IC: Current Status and Future Directions. In 2024 IEEE International Symposium on Radio-Frequency Integration Technology (RFIT) (pp. 1-3). IEEE.

[11] Shrirao, N. M. (2024, September). Intelligent design automation in VLSI systems: Leveraging AI for future electronics applications. In ECCSUBMIT Conferences (Vol. 2, No. 3, pp. 28-39).

[12] Cirstea, M., Benkrid, K., Dinu, A., Ghiriti, R., & Petreus, D. (2024). Digital electronic system-on-chip design: Methodologies, tools, evolution, and trends. Micromachines, 15(2), 247.

[13] Yin, M. (2025). Data Quality Control in Semiconductor Manufacturing through Automated ETL Processes and Class Imbalance Handling Techniques. Journal of Industrial Engineering and Applied Science, 3(6), 13-22.

[14] Islam, R. (2022). Early stage DRC prediction using ensemble machine learning algorithms. IEEE Canadian Journal of Electrical and Computer Engineering, 45(4), 354-364.

[15] Wang, J., Tse, T. K., Li, S., & Fung, J. C. (2023). A model of the sea–land transition of the mean wind profile in the tropical cyclone boundary layer considering climate changes. International Journal of Disaster Risk Science, 14(3), 413-427.

[16] Yin, M. (2025). A Data-Driven Approach for Real-Time Bottleneck Detection and Optimization in Semiconductor Manufacturing Using Active Period Method and Visualization. Academic Journal of Natural Science, 2(4), 19-26.

[17] Sabato, S. (2025). Impact of Adoption of AI in SOC Design Flow. In SOC-Based Solutions in Emerging Application Domains (pp. 173-181). Cham: Springer Nature Switzerland.

[18] Yin, M. (2025). Robust Bilevel Network-Flow Scheduling for Semiconductor Wafer Logistics under WLTP Uncertainty.

[19] Sun, Y., & Ortiz, J. (2024). An ai-based system utilizing iot-enabled ambient sensors and llms for complex activity tracking. arXiv preprint arXiv:2407.02606.

[20] Thakur, G., & Jain, S. (2025). Role of Artificial Intelligence in VLSI Design: A Review. Recent Advances in Computer Science and Communications, 18(1), E250424229315.

[21] Pang, F. (2020, November). Research on Incentive Mechanism of Teamwork Based on Unfairness Aversion Preference Model. In 2020 2nd International Conference on Economic Management and Model Engineering (ICEMME) (pp. 944-948). IEEE.

[22] Chen, Yinlei. "Daily Asset Pricing Based on Deep Learning: Integrating No-Arbitrage Constraints and Market Dynamics." Journal of Computer Technology and Applied Mathematics 2.6 (2026): 1-10.

[23] Krist, C. (2025). 17.1 Considering How to Include Human-in-the-Loop, and When to Utilize Which Tools, as Informed by the (Science) Education Literature. Applying Machine Learning in Science Education Research, 339.

[24] Pang, F. (2025). Animal Spirit, Financial Shock and Business Cycle. European Journal of Business, Economics & Management, 1(2), 15-24.

[25] Chen, Y. (2025). Generative Diffusion Models for Option Pricing: A Novel Framework for Modeling Volatility Dynamics in US Financial Markets. Journal of Industrial Engineering and Applied Science, 3(6), 23-29.

[26] Guven, I., Parlak, M., Lederer, D., & De Vleeschouwer, C. (2025). AI-Driven Integrated Circuit Design: A Survey of Techniques, Challenges, and Opportunities. IEEE Access.

[27] Manohar, R., Pingali, K., Burtscher, M., Joshi, P., & Guthau, M. (2024). An Intelligent Design Environment For Asynchronous Logic (IDEAL).

[28] Najam, L. A., & Luedkea, R. H. (2024, March). AI-Enhanced Design Automation for Next-Gen Electronics Applications and VLSI Systems. In ECCSUBMIT Conferences (Vol. 2, No. 1, pp. 47-56).