Check for updates

ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Design Reuse Via Automated Requirements-Driven Retrieval: A Framework for Electronic Hardware

Ekrem Bilgehan Uyar[1]
Graduate School of Informatics,
Middle East Technical University,
Ankara 06800, Türkiye
e-mail: uyar.bilgehan@metu.edu.tr

Cemil Gokce
Roketsan Inc.,
Ankara 06780, Türkiye
e-mail: cemil.gokce@roketsan.com.tr

Ali Ergin Gursoy
Roketsan Inc.,
Ankara 06780, Türkiye
e-mail: ergin.gursoy@roketsan.com.tr

Tugba Taskaya Temizel
Graduate School of Informatics,
Middle East Technical University,
Ankara 06800, Türkiye
e-mail: ttemizel@metu.edu.tr

Design reuse accelerates hardware development by improving efficiency, reducing costs, and shortening time-to-market. However, retrieving reusable circuit blocks directly from natural-language requirements remains challenging, particularly in confidentiality-driven low-resource domains with heterogeneous artifacts and nonstandard representations. We introduce a structured framework for requirement-driven retrieval tailored to these constraints, combining dataset curation, sparse lexical search, semantic reranking, and a large language model (LLM) for final candidate block selection. To support this, we introduce Requirements, Retrieval, Reuse: a structured electronics dataSET (R3SET) that links 338 real-world hardware requirements to 92 reusable circuit blocks extracted from eight open-source hardware projects. Each block is described using expert-reviewed, LLM-assisted metadata authoring. To simulate realistic retrieval conditions, R3SET also includes 100 synthetic distractor blocks derived from vendor artifacts. To our knowledge, it is the first dataset linking requirements to reusable electronics at this level of granularity. Evaluated on R3SET, the hybrid retriever achieves 71% Hit@1 and 92% Hit@5 on the one-to-one subset ($n = 271$), where each requirement corresponds to a single circuit block. On the full dataset including cases without a matching block ($n = 328$), the LLM slightly outperformed the retriever's top-1 accuracy (60.1% versus 58.5%) while generating decision rationales intended to support explainability. This study contributes a structured, low-resource-compatible framework for modeling requirement-to-block reuse and outlines future directions in automated input-elicitation, composite retrieval, and schematic segmentation. [DOI: 10.1115/1.4071107]

Keywords: electronic design, requirement-driven retrieval, metadata extraction, multimodal LLMs, retrieval-augmented generation, SPLADE sparse–lexical retrieval, domain-expert-in-the-loop

1 Introduction

More than an engineering cliché, "*Don't reinvent the wheel*" reflects a pragmatic stance on efficiency and serves as a cautionary principle that advocates the investigation of prior design solutions. Empirically, design reuse is a natural starting point for most designers [1], as it minimizes redundant effort using solutions that have been previously validated and optimized for performance, safety, and cost-effectiveness. In electronic hardware design, this principle is visible from the earliest stages of part selection. Device datasheets and application notes often contain "typical application circuits" that serve as templates for system architecture and schematic design.

Despite its importance, reusing hardware designs based on requirements remains difficult. Design knowledge is scattered across several design *artifacts* in inconsistent formats, making it hard to query or compare them systematically. Engineers must manually scan these, extract relevant parameters, and align assumptions before evaluating alternatives. In this context, *design artifacts* refer to a wide array of heterogeneous data representations—including models, documentation, code, design notes, and schematics—produced at various stages of the engineering lifecycle. These artifacts not only span diverse modalities and file formats but also differ significantly in structure, style, and abstraction levels due to their origins in different tools, teams, and phases of development. The resulting heterogeneity prevents designers from simply starting from a requirement and retrieving candidate blocks. Instead, they must rely on labor-intensive, error-prone manual search. A standardized, queryable repository would address this gap, but comes with significant overhead, such as structured authoring, schema maintenance, and organizational training. This motivates automation that can structure metadata, retrieve context-appropriate blocks, and generate explainable recommendations without adding heavy process burdens.

Recent advances in large language models (LLMs), especially regarding multimodality [2], create new opportunities for hardware

[1]Corresponding author.

Manuscript received October 14, 2025; final manuscript received February 7, 2026; published online April 29, 2026. Assoc. Editor: Wei Chen.

Copyright © 2026 by ASME; reuse license CC-BY 4.0

design reuse. These models can jointly process text and images, making them well-suited for domains like electronics, where key design artifacts include schematic diagrams, datasheets, and natural-language requirements. Prior work has explored the use of LLMs in requirements engineering [3–6]. LLM-augmented tools are also stated to help automate routine tasks, assist in decision making, and preserve design intent, thus reducing the likelihood of suboptimal design choices [6]. However, generic LLMs face significant limitations when applied to hardware design reuse, including handling domain-specific terminology, interpreting complex schematic data, ensuring output reliability, and addressing ethical and scalability concerns [7–11]. Their practical deployment in air-gapped or resource-constrained environments presents further challenges. These challenges highlight the need for domain-tailored, human-supervised approaches that integrate LLMs safely and effectively into engineering workflows. However, publicly available, domain-specific datasets required for training and evaluating these approaches remain scarce—a gap that is increasingly addressed through synthetic data generation and augmentation methods [12]. Recent work emphasizes that dataset creation efforts must move beyond ad hoc practices toward principled, systematic methodologies [13], especially for emerging applications such as hardware design reuse.

This study addresses these challenges by proposing a requirement-driven retrieval framework that reduces the cost and friction of structured reuse in hardware design. We focus on low-resource, cloud-independent settings where data scarcity, confidentiality, and computational limits preclude conventional cloud-based artificial intelligence (AI) solutions. Our work is guided by four research questions:

RQ1: How should a dataset be designed for developing and evaluating requirement-driven reuse systems in data-scarce environments?

RQ2: To what extent can reusable circuit blocks be reliably retrieved from natural-language requirements using a low-resource, cloud-independent system?

RQ3: How does a resource-constrained LLM-based decision layer behave when selecting circuit blocks from retrieved candidates?

RQ4: Does requirements-driven retrieval of reusable circuit blocks reduce design time and perceived effort without degrading expert-rated correctness?

To answer these questions, we first defined the metadata necessary for requirement-driven queries and authored structured guidelines for expressing it. Using these guidelines, we applied multimodal LLMs (MLLMs) to extract metadata from real-world circuit images, iteratively refining outputs through expert review. This process yielded a validated, curated dataset. We then designed a lightweight offline retrieval pipeline using Sparse Lexical and Expansion Model (SPLADE) for sparse search and Qwen2.5 for LLM-based ranking and selection. This study contributes the following:

- Expert-Guided Metadata Modeling Framework: A structured process model for constructing and validating requirement-block datasets using MLLMs and iterative expert supervision, defining reproducible metadata extraction, labeling, and consistency verification loops.
- *Requirements, Retrieval, Reuse: A Structured Electronics Dataset (R3SET)*: The first open-access dataset linking natural-language hardware requirements to reusable circuit blocks. R3SET enables controlled, reproducible comparison of requirement-to-block retrieval and selection methods under a fixed corpus, candidate pool (including synthetic distractors), and evaluation protocol, with explicit coverage of one-to-one and no-match (NM) cases.
- Hybrid Sparse-Semantic Retrieval Framework: A domain-adapted retrieval pipeline that fuses SPLADE-based lexical expansion and BGE-M3 semantic embeddings using unified scoring and reranking. It defines how sparse and dense signals can be systematically combined and evaluated to support requirement-driven circuit reuse in low-resource engineering settings.

Through this work, we provide both empirical evidence and actionable insights into how generative AI can support structured design reuse. Our findings highlight the technical and organizational conditions necessary to integrate LLMs into hardware design workflows reliably and at scale.

2 Related Studies

Design reuse helps hardware teams save time and reduce risk using tested subsystems. However, reuse is difficult because design knowledge is scattered across datasheets, schematics, code, and documents without a unified structure [14,15]. Integrating these sources manually is slow and error-prone, and trust issues with unofficial data discourage reuse [16]. Engineers cannot simply search for requirements in plain language to find relevant blocks and instead must navigate fragmented files.

Recent advances in generative AI have shown promise in automating electronics design tasks [17,18]. Most of the existing work focuses on integrated circuit design, but similar concepts can be applied to higher abstraction levels, such as printed circuit board (PCB) functional blocks. A major challenge in this integration is the domain-specific nature of the hardware. Circuit data are highly heterogeneous, and even small structural changes can lead to significant differences in behavior, making modeling and automation difficult [18]. This has two main implications for design reuse. First, design artifacts must be captured and stored in a way that is searchable and semantically meaningful to support effective retrieval. Second, reuse should focus on higher-level functional blocks rather than individual components. By working at the block level, systems can avoid dealing with minor structural variations directly, simplifying both reuse and the reasoning needed to match designs to requirements.

To our knowledge, there exists no publicly available dataset that links natural-language hardware requirements to reusable circuit blocks in a format suitable for automated retrieval. Existing studies mainly address software traceability [19], hardware description language (HDL) or text datasets [20,21] that are not and cannot be aligned with requirement-to-block reuse. Other efforts like diagram-to-netlist conversion [22,23] or generative design pipelines [24] do not provide ground-truth mappings for reuse. While the importance of dataset quality and diversity for electronic design automation (EDA) is emphasized, it is also noted that such datasets are scarce due to proprietary restrictions and the lack of openly accessible contributions [25]. However, building such a dataset is difficult since it must capture a diverse set of circuit functions while keeping descriptions consistent [26]. Diversity ensures different blocks can be distinguished, while consistency keeps features aligned with their sources. A requirements dataset [27] that focuses specifically on natural-language hardware requirements extracted from eight open-source designs partially addresses this gap, providing an initial foundation for studying requirement-level representations in the hardware domain. However, restricting the functional block dataset to components drawn only from the original eight projects still risks introducing a *closed-world bias*, which can artificially inflate retrieval performance by limiting diversity and concealing real false positives and false negatives.

Similar challenges have been discussed in the natural-language processing (NLP) domain. A recent study [28] highlights the value of incorporating distracting passages when evaluating retrieval-augmented generation (RAG) systems, showing that passages with higher distracting effects reduce accuracy and expose system weaknesses. Another study [29] emphasizes that RAG systems must handle ambiguous queries and potentially conflicting information while suppressing inaccurate or irrelevant documents. To address this and rigorously evaluate *RQ2*, we adopted the

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

dataset proposed in Ref. [27] and augmented it with synthetic distractor entries, thereby introducing structured noise that better reflects real-world retrieval scenarios. Similar strategies have been adopted in the NLP domain [30,31], where synthetic data are generated to improve the performance of their RAG system.

Current AI tools, particularly generic foundation models, often misinterpret circuit data, misread interactions, or add incorrect details. Observations from a recent hardware analysis study [32] evaluating GPT-4o indicate that while these models demonstrate competence in handling formulas and recognizing circuit elements, they exhibit inconsistent performance when interpreting netlists. These inconsistencies are often not the result of arithmetic limitations, but rather arise from misinterpretations of component interactions, spatial orientations, contextual dependencies, and, in some cases, hallucinated configurations or overlooked instructions. In hardware design reuse scenarios, such limitations are further compounded by the presence of domain-specific terminology and the inherent complexity of engineering data, which often leads to misinterpretation of schematic symbols, hallucination of components, and ultimately, a lack of reliability. These concerns highlight the need for either domain-specific model adaptation or human-in-the-loop oversight.

Retrieval for design reuse has recently gained attention, with RAG emerging as a key approach to combine structured retrieval with language models for improved accuracy and interpretability in electronics design. For instance, HSG-RAG (Hierarchical Semantic Graph Retrieval Augmented Generation) [33] focuses on knowledge base construction for embedded system development, extracting information from large technical document collections and organizing it into a hierarchical structure to improve retrieval efficiency and precision. It also introduces a benchmark dataset comprising question-answer pairs that were built from three hardware platforms and three peripheral devices. Another study [34] explores the use of local language models with RAG for PCB design tasks, integrating natural-language interfaces with electromagnetic simulation tools to allow nonexpert users to design and simulate components such as differential microstrips and strip-lines. Bai et al. [35] present VerRAG, a RAG-enhanced framework for RTL (Register-transfer level) code generation. They propose a RISC-V-based co-processor architecture template and a Summary-Template RAG (ST-RAG) for template-driven RTL generation, showing its effectiveness in producing AI-specific co-processors with performance benefits. Similarly, EDA-adaptive RAG [36] introduces a low-cost framework for EDA tasks. It includes a comprehensive RTL-to-GDSII corpus generation and processing method and uses a query classifier with an improved retrieval optimizer to dynamically narrow search scopes. These approaches show that combining structured retrieval with targeted generation can improve efficiency, precision, and accessibility in hardware design workflows. Unlike recent studies, our study creates a curated dataset linking natural language hardware requirements to reusable circuit blocks and develops a retrieval pipeline capable of handling both lexical and semantic matching for candidate block selection.

Even with an ideally constructed dataset for a requirement-driven circuit reuse system, traditional sparse retrieval methods, which rely on exact-term matching, struggle with complex linguistic phenomena such as *semantic similarity, synonyms, and polysemy*, limiting their ability to capture deeper semantic relationships or overcome vocabulary mismatches effectively [37,38]. In contrast, learned dense retrieval models are generally better at computing *semantic similarity* and *generalizing across related concepts* [39]. Therefore, hybrid approaches are often required for retaining exact-term matching while incorporating semantic expansion to overcome vocabulary mismatches [40]. Advanced approaches such as knowledge graphs [41] are proposed as an effective way to organize and systematically structure this heterogeneous and fragmented data, but we pursue a simpler, lightweight strategy that normalizes this information into plain structured text. For this purpose, we adopt sparse lexical search in the first stage, contrasting with purely dense vector retrieval. Neural sparse approaches like SPLADE produce compact representations while achieving accuracy comparable to dense embedding methods [42].

3 Methodology

The proposed methodology is designed to address the research questions and included three main phases as illustrated in Fig. 1: (1) the curation of the dataset, (2) the design of a retrieval mechanism to identify candidate circuits, and (3) the development of an LLM-based module to determine the final match using both the requirements and the candidates retrieved.

3.1 Dataset Requirements and Base Dataset. The proposed methodology requires a well-structured corpus to evaluate retrieval performance. However, no publicly available dataset currently links natural-language requirements with circuit blocks in a structured, queryable format suitable for generative AI-based retrieval. To address this gap, we defined a set of criteria to guide the creation of a realistic, high-quality dataset.

First, all source materials must be drawn from open-source hardware projects to ensure reproducibility and free redistribution. The selected projects should span a broad range of application domains to capture different hardware components and functions. Diversity is also needed in project structure, representation formats, and documentation quality to reflect the variability found in real-world design environments. Requirements should be authored by multiple authors to provide natural variation in language, style, and specificity.

Second, each requirement must be directly linked to the physical hardware design elements it describes, enabling accurate evaluation of retrieval performance. The dataset should include a mix of realistic matching scenarios, including one-to-one matches, one-to-many matches (a requirement mapping to multiple blocks), and cases with no valid match. Finally, every circuit block should be represented with a minimal yet extensible metadata schema that captures essential attributes such as function, inputs and outputs, operating constraints, and distinguishing components, while remaining compatible with LLM-driven retrieval methods.

Since building a dataset from scratch is both time-intensive and resource-demanding, we first evaluated whether an existing dataset could be repurposed for our task. For this purpose, we selected an existing corpus of 338 low-level hardware requirements curated from eight open-source electronics projects [27]. This corpus was developed through a systematic methodology to support requirement classification tasks. The curation and labeling of this dataset followed the MATTER cycle (model, annotate, train, test, evaluate, revise) [43], a structured, iterative framework recognized in the requirements engineering community. This process ensured a reliable hardware requirements dataset, providing a solid foundation for extension and adaptation to our retrieval task.

Originally developed for a classification task, the dataset was designed to assign each requirement to one of five functional categories: controller, driver, feedback, interface, or power. These map to canonical roles in embedded electronics—controller (compute/control logic), driver (actuation and signal drive stages), feedback (sensing and monitoring), interface (external communication and user I/O), and power (supply, regulation, and delivery)—each carrying distinct linguistic signatures. These common subsystem roles in embedded hardware can form the basis for semantic retrieval and block matching.

The projects, as can be seen in Table 1, span diverse domains including robotics, agriculture, and energy systems, ensuring broad coverage rather than concentration in a single application area. Table 2 summarizes the distribution of requirements across functional categories, projects, and authors. To support detailed performance analysis and error diagnosis, each requirement was

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

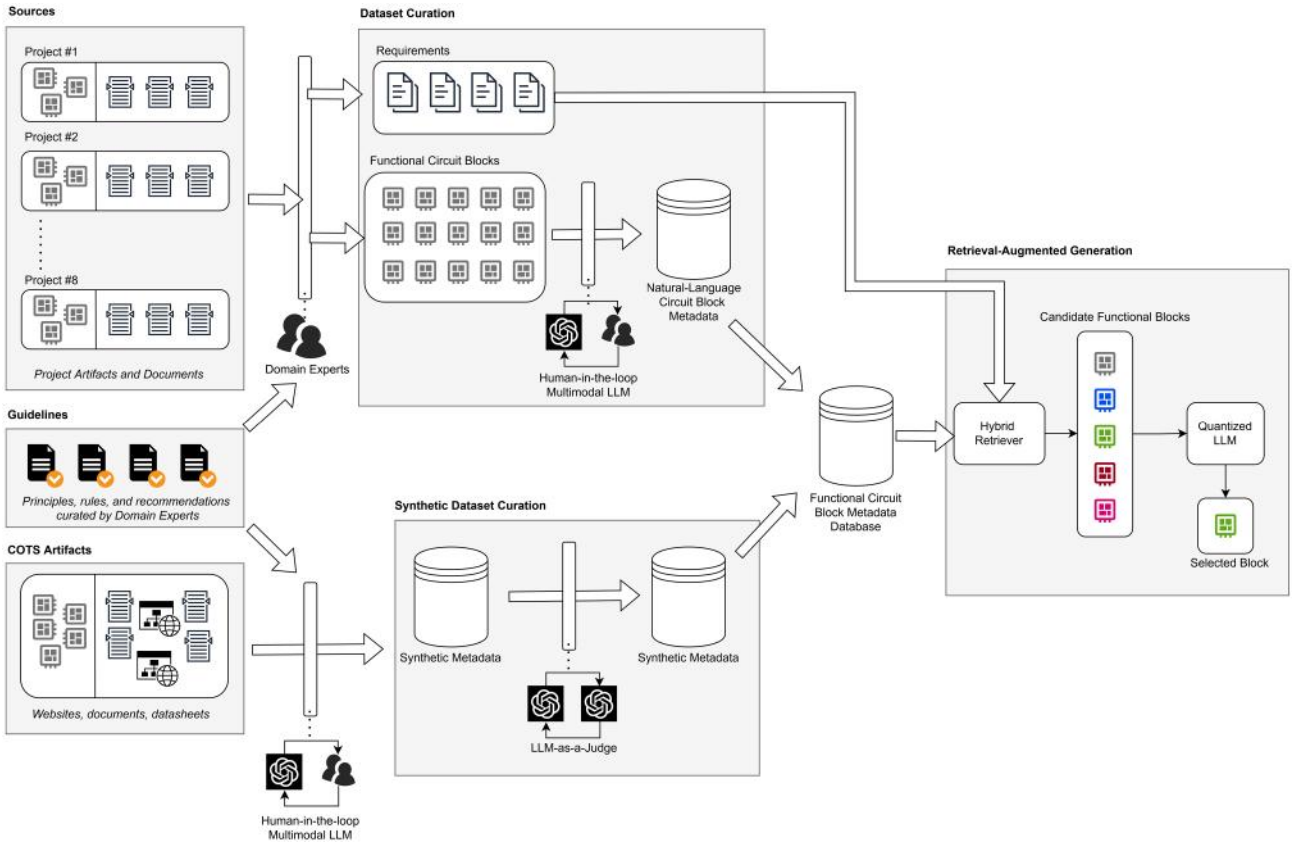

Fig. 1 Overview of the study

annotated with its original author, project source, and functional category. This structuring enables stratified evaluation by phrasing style, project domain, and circuit function, all of which impact retrieval behavior.

The category distribution is reasonably balanced, with no single class dominating the corpus; notably, interface and feedback requirements are the most frequent, reflecting typical real-world emphasis on connectivity and sensing. Project-level variation reflects differences in requirement density but avoids extreme imbalance. The dataset was authored by three domain experts (DEs), whose distinct expression styles introduce lexical and syntactic variation, which is important for assessing generalization in retrieval performance.

Table 1 Functional domains of projects

ID	Functional domain
P001	Generic embedded control platforms: a flexible IoT controller board applicable across multiple use cases [44]
P002	Robotics: control system for a hydraulically actuated quadruped [45]
P003	Additive manufacturing: multifunction extruder with integrated process sensing and monitoring [46]
P004	Food processing and preservation: automated control for beef dry-aging [47]
P005	Actuator control: high-current drive electronics for magnetic actuators [48]
P006	Agricultural automation: sensor-driven irrigation scheduling and control [49]
P007	Environmental monitoring: autonomous ocean profiler for hazardous-water measurements [50]
P008	Power electronics: modular controller for a voltage-source inverter [51]

The selected corpus (hereafter called *base dataset*) reflects the informal and heterogeneous nature of real-world engineering documentation. Some requirements are ambiguous or incomplete, consistent with the way engineers typically capture design intent during early development. While such noisiness makes the dataset less suitable for formal verification or large-scale supervised training, it provides a realistic testbed for retrieval. In practice, design engineers are unlikely to provide perfectly structured requirements, so evaluating in imperfect, unstandardized language better reflects actual usage conditions and more rigorously challenges the retrieval and ranking process.

Table 2 Functional category, source project, and author distributions

		Count	Percentage
Functional category	Controller	30	9.15%
	Driver	54	16.46%
	Feedback	77	23.48%
	Interface	95	28.96%
	Power	72	21.95%
Source projects	P001	36	10.98%
	P002	41	12.50%
	P003	36	10.98%
	P004	35	10.67%
	P005	48	14.63%
	P006	59	17.99%
	P007	43	13.11%
	P008	30	9.15%
Domain experts	DE1	100	30.49%
	DE2	106	32.32%
	DE3	122	37.20%

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Collectively, these characteristics indicate that the corpus is sufficiently diverse and realistic to be used and extended for evaluation of retrieval performance. While its size is adequate for this purpose, it is too small to train modern large-scale models. The corpus is suitable for use as the natural-language input to our retrieval system, but it has two important limitations: first, it was originally developed for a classification task rather than retrieval, and second, it does not include links to the physical hardware designs described by the requirements.

3.2 Expert-Guided Metadata Modeling Framework: Building a Requirements-Linked Functional Block Database. To evaluate the proposed generative AI-based retrieval approach for hardware redesign, we extended the *base dataset* into a structured, requirements-linked functional block database. This new extended dataset, R3SET, aims to enable effective and scalable search over existing circuit blocks, which is essential for retrieval-driven reuse.

The reusable blocks in our dataset are manually derived from multimodal and unstructured source project documents, i.e., schematics, datasheets, and informal descriptions, which are not themselves included in the base dataset. To enable querying, each block is described using a minimal, extensible metadata schema designed for both human interpretation and LLM-based retrieval. The schema focuses on four key descriptive fields necessary for reuse decisions: (1) block function (the purpose of the circuit), (2) I/O behavior (interfaces and signals), (3) constraints (operational or design limitations), and (4) distinguishing components (critical elements differentiating similar blocks). It should be noted that these metadata fields are specific to circuit blocks and would need adaptation for other domains, although the expert-guided modeling process is domain-agnostic.

Creating accurate metadata is time-consuming and error-prone if done entirely manually. To address this, we adopted a human-in-the-loop workflow where DEs oversee and validate outputs, while MLLMs assist in generating and refining metadata descriptions. The three DEs—electrical and electronics engineers with over 10 years of industry experience in embedded systems—voluntarily participated in these processes. While technically proficient, none are native English speakers, which adds to the language variation.

To support realistic retrieval evaluation, the base dataset was extended to include explicit requirement-to-block links. These links reflect three real-world matching scenarios: (1) one-to-one matches, where a requirement corresponds to a single block, (2) one-to-many matches, where a requirement maps to multiple related blocks, and (3) no-match cases, where no block satisfies the requirement.

Finally, to increase task difficulty and simulate real-world noise, *distractor* blocks from external sources are added. These blocks are structurally valid but unrelated to any valid requirement, ensuring that the retrieval system must discriminate between relevant and irrelevant candidates.

The rest of this section describes the metadata creation procedure, manual block extraction workflow, LLM-assisted metadata generation, and the procedures used to evaluate metadata quality in detail.

3.2.1 Metadata Design and Schema Extraction Guideline Development. The heterogeneity of design artifacts across source projects ranging from informal schematics to incomplete documentation necessitates a structured, repeatable workflow for extracting reusable circuit blocks and representing them in a consistent, queryable format.

The process begins with defining a domain-specific metadata schema that captures only the essential information required for retrieval based on natural-language requirements. The schema must be general enough to apply across diverse electronic subsystems, yet specific enough to distinguish blocks by function, behavior, and key components.

Table 3 Required metadata fields

Column name	Description
BlockID	001…999, matches image filename
ProjectCode	Same short code on schematic title block
Category	F / C / I / P / D [27]
FunctionalName	Human-readable, unambiguous name ≤12 words (e.g., *3-phase current sense frontend with opamp gain stage*)
DetailedFunction	Full-sentence description of purpose and principle ≤50 words (e.g., this block senses phase current via a low-ohmic shunt and amplifies the voltage with a differential Op-Amp for Analog-to-Digital Converter input)
InputOutputSummary	Plain language I/O spec: signals in/out, levels, reference (e.g., inputs: 0–10 A phase current → output: 0–3.3 V analog voltage to microcontroller Analog-to-Digital Converter)
KeyComponents	List of core active/passive part types (e.g., shunt resistor, opamp, low-pass filter, pull-down resistor)
Constraints	Voltage, isolation, electromagnetic interference, speed, or accuracy constraints (e.g., operates from 24 V rail, bandwidth 10 kHz, nonisolated, low offset required)
SimilarityTags	Comma-separated list for retrieval (e.g., current sense, analog frontend, opamp, shunt, feedback block)
BriefDesc	Full-sentence description of purpose and principle ≤25 words
OriginalCoords	Upper-left + lower-right schematic grid refs, e.g., "B2-D4." Always report rectangular bounding box on a single sheet. Split across pages requires multiple Grid Coords.
Author	Initials or anonymized unique ID (e.g., "DE01")

We implemented a two-stage approach. First, we designed a practical, minimal schema (Table 3) aligned with the way circuit blocks are treated in system-level design: each block is defined by its function, inputs/outputs, and operational constraints. To support diverse retrieval needs, the schema also includes category labels for functional grouping, key components for part-level queries, and expert-curated similarity tags for lexical matching. A brief description field enables quick filtering, while project identifiers, source coordinates, authorship, and a unique BlockID ensure traceability and reproducibility. Second, we applied this schema in a human-in-the-loop curation process, where the DEs systematically extracted blocks from source projects while MLLMs assisted in generating and refining metadata. This ensured accuracy and reproducibility while reducing the manual burden on experts. The proposed schema is language-model compatible and can be extended to include additional fields if required (e.g., testability or reliability metrics, optimization parameters for PCB layout, and environmental specifications).

To ensure consistent and reproducible metadata across all projects, we developed a guideline for applying the proposed schema. The goal of this guideline is to standardize heterogeneous hardware design artifacts into a unified library of circuit blocks. Each block is described using normalized, technically accurate natural-language descriptions that support both human understanding and LLM-based retrieval. The guideline is organized into two main tasks: (1) block extraction defines how to segment complex, heterogeneous schematics into self-contained functional blocks. (2) Metadata authoring specifies how each extracted block is described using technically accurate, normalized terminology. The process enforces strict field definitions and value formats (see Table 3).

The *block extraction* task applies a systematic process for segmenting schematic sheets into self-contained functional block

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

images. Each extracted block includes all relevant pins and nets, retains contextual labels, and records the source project reference for traceability. To maintain consistency across diverse projects, three inclusion criteria were applied:

(1) *Functional Coherence*—The block must perform a self-contained function (e.g., analog signal conditioning and power conversion).
(2) *Reuse Potential*—The block must be general enough to plausibly recur in other designs.
(3) *Extractability*—The block must be physically and logically extractable without ambiguity or dependency on undocumented context.

Additionally, exclusion rules were provided to avoid clutter and irrelevant data. For example, trivial passive-only nets or vendor-specific mechanical components were omitted, as they do not contribute to functional reuse.

The guideline also covers: (1) resolving naming inconsistencies across projects, (2) handling varying abstraction levels in schematics, (3) writing clear, technically accurate descriptions that are accessible to both engineers and LLMs.

By following this two-part process, the resulting library provides a standardized, high-quality foundation for requirement-driven circuit-block retrieval.

3.2.2 Manual Circuit-Block Extraction. Since these open-source projects were independently developed for specific applications rather than reuse, their schematics lack modularity, showing low cohesion and high coupling. Since the schematics and design files lacked explicit delineations of functional blocks, expert interpretation was required to identify and extract reusable circuit blocks. This process involves understanding both the technical function of each circuit and the broader design context in which it appears.

Although automation techniques such as rule-based parsing, template matching, and early-stage vision-language models were considered, manual extraction was ultimately preferred due to the heterogeneity of the schematic formats, the absence of validated segmentation models, and known limitations [32] of the state-of-the-art models in circuit understanding. Therefore, following the defined workflow, the DEs identified functional blocks and schematic sections representing isolated functional blocks.

During extraction, the schematics were occasionally disaggregated to separate overlapping blocks, and inconsistencies in names were resolved to ensure clarity. Component-level notes and bills of materials were cross-checked to validate each block's intended functionality. Trivial or nonfunctional elements, such as logos or mechanical mounting holes, were excluded.

The initial pool of 97 candidate blocks was extracted by one DE, while the other two independently reviewed the blocks and flagged 11 of them for further discussion. A consensus meeting was held, during which all three experts jointly reviewed the flagged items. Decisions were made by unanimous agreement: four blocks were split due to containing multiple distinct functions, five were excluded for lacking reuse value or being trivial sensor interfaces, and two were excluded for duplication. This resulted in a finalized set of 92 blocks.

3.2.3 Large Language Model-Assisted Metadata Creation From the Extracted Circuit-Block Images. The objective of this phase is to represent reusable circuit blocks using a structured metadata schema to facilitate retrieval, minimizing manual authoring effort while ensuring technical consistency. To achieve this, we adopted a human-in-the-loop approach, combining the generative capabilities of general-purpose MLLMs with expert oversight. As no specialized models were available to us, we used closed-source, commercial, general-purpose MLLMs (GPT-4o and GPT-5) as drafting assistants.

The assistant was prompted (see Appendix A) with schematic images and instructed to generate structured metadata following a predefined schema (Table 3) and accompanying guideline. It was tasked with (1) identifying the block's primary function and I/O behavior, (2) extracting key components with meaningful, context-aware names, and (3) populating other fields such as operational constraints and distinguishing characteristics.

We implemented a schema compliance check with the DEs and iteratively refined the LLM prompts. Common corrections involved replacing generic designators (e.g., "U1") with descriptive net or function names (e.g., "3V3_RX"), enforcing standardized units and terminology, removing nontechnical or marketing language, completing missing parameters using datasheet references, restricting component identifiers to the KeyComponents field while prioritizing the primary function for multipurpose blocks, and, for controller-type blocks, explicitly documenting processor type, memory size, interface protocols, and peripheral counts.

Metadata entries for each of the 92 functional circuit blocks were returned. These outputs were then systematically reviewed by DEs to verify technical accuracy, completeness, and adherence to the schema before being integrated into the final requirements-linked block database. The verification step and quality checks are described in the next section.

3.2.4 Evaluating Metadata Quality. To ensure the reliability of the generated metadata, each circuit-block entry was evaluated across five dimensions: *consistency*, assessing whether similar information is expressed in a uniform format; *completeness*, ensuring all required fields are populated; *accuracy*, verifying alignment with the circuit's actual function and datasheet specifications; *retrievability*, determining whether the metadata supports effective retrieval in RAG or LLM-based systems; and *clarity*, confirming that entries are understandable to human engineers without ambiguity.

Among the five dimensions, *completeness* is already satisfied, as the assistant produced no unpopulated rows. We determined that *retrievability* should be assessed within the *retrieval pipeline*, and the remaining three dimensions can be evaluated reliably only under *human oversight*. Consequently, addressing these dimensions, the DEs conducted a review of each entry, focusing on three specific criteria for this human-in-the-loop evaluation framework:

(1) adherence to the metadata guideline,
(2) coherence between the circuit block's image and its textual description, and
(3) correctness of the specified technical parameters with respect to technical paper, circuit topology, and datasheets.

Flagged entries were resolved through consensus by either *revising* specific fields, fully *replacing* their content, or completely *removing* the associated circuit block. *Revisions* and *replacements* addressed factual inaccuracies, vague wording, or misinterpretation of functionality, often stemming from LLM hallucinations. *Removal* was reserved for cases of structural inconsistency or guideline violation that could not be resolved with edits.

As a result, eight blocks were removed entirely, and 34 were revised (either partially or fully) before inclusion in the final metadata set.

3.2.5 Requirement-Block Mapping and Annotation. This step is used to map the *base dataset* requirements with the functional circuit-block metadata that satisfy them. The mapping process was performed manually by the three DEs with consensus meetings. Each requirement was assessed against all blocks in the library to determine whether a match existed and in what form. The annotation scheme accommodates one-to-one (1:1) and one-to-many (1:M) mappings, but excludes many-to-one (M:1) and many-to-many (M:M) mappings:

- If a requirement is satisfied by a single block, that block's ID is directly assigned (1:1).

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

- If multiple blocks independently fulfill the requirement (logical OR), each block is listed as an alternative match (1:1).
- If a requirement is satisfied only when multiple blocks are used together (logical AND), their IDs are recorded jointly (1:M).
- If no block satisfies the requirement on its own—either due to abstraction level or absence from the block library—the requirement is labeled as NM.

M:1 and M:M mappings were excluded to simplify evaluation and focus on retrieval precision at the individual requirement level. In practice, multiple requirements may collectively refer to a single functional block, especially when requirements are written in a fragmented style, e.g., specifying measurement range and resolution separately instead of as a unified capability. Accommodating such mappings would require composite interpretation and clustering of semantically related requirements, which we leave for future work.

No-match cases typically arise when a requirement refers to a high-level system function, ambiguous requirements, M:M or M:1 type requirements, or when the original design uses commercial off-the-shelf (COTS) modules not represented in the extracted block set.

As a result of this process, out of 338 requirements, 271 were mapped to a single block (1:1), 10 were assigned to multiple blocks (1:M), and 57 were labeled as nonmatchable (NM) due to scope, ambiguity, or abstraction level. We additionally performed a context-independent second pass over all requirement-block mappings (requirements in isolation), updating 56 assignments, and we report results based on this revised mapping. From guideline preparation to final consensus meetings, the entire process took 4 months and required approximately 294 total DE-hours.

3.3 Synthetic Metadata Curation With Expert-Guided Multimodal Large Language Models. The synthetic circuit-block metadata are introduced as adversarial test cases to stress-test retrieval precision under realistic noise. Their role is to mimic real metadata and serve as distractors that reveal retrieval weaknesses, following standard RAG evaluation practices in the literature. Unlike the validated design blocks, they only need to be lexically plausible and structurally consistent—not technically correct.

This phase employed the same human-in-the-loop, multimodal LLM workflow described earlier but with modified inputs and prompting strategies. Instead of schematics from the original projects, the assistant generated metadata from external sources, such as datasheets, vendor websites, evaluation board manuals, and device documentation. When canonical application circuits were unavailable, the assistant was prompted (see Appendix A) to infer plausible usage scenarios, producing metadata as though the block had been implemented.

The following subsections detail the inclusion criteria, generation process, and evaluation of synthetic metadata quality.

3.3.1 Inclusion Criteria. We augmented the curated dataset with 100 synthetic blocks, matching the original block count to simulate a balanced retrieval pool. These were sampled from vendor catalogs filtering components based on voltage level, interface, and functionality metadata. Then they were classified into off-domain (80%) and in-domain (20%) categories. Off-domain blocks originate from functional categories absent in our source projects (e.g., radio-frequency transceivers, audio codecs), while in-domain blocks share high-level roles (e.g., switching regulators, I/O expanders) but differ in implementation details.

The 80–20 split reflects a realistic retrieval distribution in which most candidates are irrelevant, while a small fraction serves as high-confusion distractors. This split also enables a calibration study on 20 in-domain synthetic blocks where the DEs could verify metadata against their domain knowledge, while the remaining 80 off-domain blocks represent functions outside the DEs' primary expertise, making direct expert validation speculative rather than authoritative. This split also enables stress-testing of both the retrievers and the LLM-based selectors. Off-domain entries were selected from widely used embedded system components with high availability, such as pH sensors, audio signal processors, advanced spectral detectors, and active cooling controllers. These devices share common interfaces or formats (e.g., communication protocols like SPI, differential input, 5V supply) but differ in purpose and context. Their inclusion broadens both the functional and lexical diversity of the dataset, ensuring the retrieval system is tested beyond narrow matching scenarios. In contrast, in-domain entries were deliberately selected to closely match the target blocks in topology, component types, and terminology, while differing subtly in functionality. For example, a near-miss distractor for a temperature-sensing block might include a similar analog front-end but be designed for pressure instead. These cases require the system to distinguish fine-grained functional intent despite high structural overlap.

This phase concluded with the manual identification of all the candidate devices by scanning large-scale online component distributors' inventories, gathering technical documentation, and organizing artifacts by type (e.g., datasheet, demo board, and product page). The collection comprises 97 unique products from 24 distinct manufacturers. The product portfolio spans diverse categories including various sensor types (optical, acoustic, touch, inertial measurement unit, and radar), audio/power management integrated circuits, wireless communication modules, microcontrollers, data converters, and memory devices, collectively representing a comprehensive set for modern embedded systems and IoT applications. Selection emphasized overlap in terminology with the main dataset while introducing new tokens, structures, and roles to test generalization. These artifacts then served as inputs for generating tailored prompt sets in the next step.

3.3.2 Synthetic Metadata Creation. To generate synthetic entries consistent with the real metadata schema, we employed an MLLMs assistant (GPT-4o and GPT-5) guided by a structured prompt (see Appendix A). Unlike the original dataset, which was derived from circuit-block images, these entries were created from publicly available sources such as datasheets, reference designs, evaluation board manuals, and vendor websites.

When application circuits were present, the assistant directly used them to generate metadata. In their absence, it was instructed to infer plausible usage scenarios and produce metadata as if the block had been implemented. All outputs adhered to the existing authoring guideline, avoiding promotional language, and emphasizing technical clarity and quantitative detail.

Each synthetic record followed the exact structure of the real dataset. Trade names were restricted to the KeyComponents field, while all other fields required generic descriptions. The assistant was prompted (see Appendix A) to include operating limits, interface protocols, and distinguishing constraints to ensure comparability with the original entries.

Postprocessing was conducted to enforce consistency, correcting errors such as mislabeled block types, refining constraint phrasing, and standardizing column order. Synthetic entries were assigned reserved identifiers (e.g., BlockID = Mxxx, ProjectCode = DGKY) and explicitly marked as nonvalidated. Each record also included provenance metadata, including vendor name, source document type, version, and URL.

These corrections ensured uniformity and traceability but did not imply validation of technical accuracy. The final outcome was a set of 100 synthetic entries, complete with source records and input artifacts, ready for use in retrieval evaluation.

3.3.3 Evaluating Synthetic Metadata Quality. To assess and improve the quality of the structured metadata associated with synthetic distractor blocks, we implemented an automated *LLM-as-a-Judge* strategy based on GPT-4o. This approach was inspired by recent work using LLMs for synthetic data quality

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

assessment in retrieval pipelines, particularly for RAG evaluation tasks [52].

Our strategy followed a rule-augmented prompt design (see Appendix A). For each synthetic block, the LLM received (i) the metadata row, (ii) the original datasheet of the block, and (iii) the metadata authoring guidelines used during manual annotation. The prompt instructed the model to act as an expert hardware design reviewer and return (a) a binary quality verdict (*ACCEPTABLE* or *NEEDS IMPROVEMENT*), (b) a concise justification ($\leq$3 sentences), and (c) targeted field-level correction suggestions.

To reduce the risk of systematic errors when using an LLM as a metadata judge, we performed a calibration on a subset of the synthetic blocks. From 100 synthetic blocks, we selected 20 "in-domain" blocks for which the DEs were comfortable assessing internal consistency. DEs reviewed these 20 blocks using the same criteria as the judge prompt, noting only minor deviations that would not materially change retrieval behavior. We then applied the LLM-based judge to the same 20 blocks. In 19/20 cases, the judge accepted the metadata without changes; in one case, it suggested a minor adjustment to the constraints of a hypothetical circuit, which the DEs also considered noncritical. We then applied the judge to the remaining 80 off-domain blocks, where attempting direct expert validation would have been speculative (DEs cannot confidently assess). However, because the judge demonstrated reliable conformance checking on in-domain cases, and because technical correctness is not required for distractor efficacy, we used the judge as a consistency filter rather than a technical validator. Eleven of 100 entries were flagged for guideline violations (e.g., trade names in function fields, vague constraints). The DEs reviewed these and confirmed they represented minor deviations that would not materially affect retrieval behavior, so they were retained.

3.4 Hybrid Sparse-Semantic Retrieval Framework for Requirement-Driven Hardware Reuse. To operationalize retrieval over the R3SET dataset, we designed and developed a retriever system that maps natural-language requirements to relevant circuit blocks using both lexical and semantic signals. The dataset presents several challenges: requirements vary widely in phrasing, granularity, and style, while block metadata is structured but still contains noise. Exact matching is rarely sufficient, and semantic cues must be extracted without losing alignment with functional terms. The proposed system addresses this by combining sparse term-weighted retrieval with dense semantic embeddings, jointly indexed in a hybrid vector database. Given a requirement, the system performs parallel encoding, fused scoring, and optional cross-encoder reranking.

The following subsections detail the rationale behind this design, the construction of indexable metadata, the query and scoring pipeline, and the evaluation setup used to assess retrieval performance.

3.4.1 Retrieval Problem and Design Rationale. The retrieval task is to identify and rank the most relevant circuit blocks for a given natural-language requirement. Each block in the dataset has structured metadata fields intended to capture its operational role and implementation context. However, the linguistic mismatch between informal, high-level requirements and formalized metadata descriptions makes retrieval nontrivial. Requirements may contain vague references, implicit assumptions, or domain-specific phrasing that do not surface as direct lexical overlap with the metadata. For example, each requirement is generally a standalone sentence, often containing implicit constraints or function specific terminology, e.g., "The system shall support simultaneous two-way communication with external world." The target blocks, by contrast, are described using structured metadata fields that vary in length, vocabulary, and specificity. For example, one block may specify "Full-duplex RS-485 interface", while the requirement refers only to the need, "simultaneous" and "two-way", without naming exact terminology.

This lexical and structural mismatch prevents simple string matching or keyword search from achieving acceptable precision. Some requirement terms may not appear verbatim in metadata but still imply strong relevance (e.g., "Bluetooth LE" versus "BLE module supporting 2.4 GHz"). Conversely, metadata may contain technical detail irrelevant to the requirement, which can dilute similarity scores.

Accordingly, to address *RQ2*, we designed a retrieval system that bridges these gaps. The system is designed around two complementary needs: (1) preserving sparse lexical cues that may be important for functional disambiguation (e.g., "stepper motor," "RS-485"), and (2) enabling fuzzy semantic matching to account for variation in wording and abstraction (e.g., "digital interface" versus "UART"). This motivates the use of a *hybrid retrieval architecture*.

Retrieval must be performed offline, over a few hundred blocks, on local hardware-requiring efficient index structures and low runtime overhead. Therefore, the vector store should support hybrid sparse–dense indexing and compatibility with both embedding types without cloud dependency.

The system must return top-k candidate BlockIDs for each query, optionally refined by a cross-encoder reranker. The following subsections detail how metadata and queries are processed, how scoring is computed, and how retrieval quality is evaluated.

3.4.2 Metadata Processing and Index Construction. To balance descriptive fidelity with generalizability, the retriever operates over a corpus of candidate blocks, each annotated with four structured metadata fields: `FunctionalName`, `DetailedFunction`, `Constraints`, and `SimilarityTags`. `FunctionalName` provides a short role label (e.g., "RS-485 Transceiver"); `DetailedFunction` elaborates specific behaviors (e.g., "supports full-duplex differential transmission"); `Constraints` encodes electrical or protocol-level requirements (e.g., "up to 10 Mbps"); and `SimilarityTags` captures implicit grouping cues, often drawn from prior reuse patterns (RS485, RS 485, and RS-485) or functional taxonomies (e.g., "interface," "communication," and "serial"). These fields were selected because they are both structurally consistent across blocks and semantically rich enough to support retrieval from underspecified queries. Fields such as `BlockID`, `ProjectCode`, or designator-level attributes were excluded from indexing due to their identifier-like nature or limited retrieval value.

To prepare block metadata for retrieval, we concatenated four descriptive fields into a single text string. Each field is prefixed with its label (e.g., "Name:," "Constraints:," and "Tags:") to preserve semantic boundaries. This format enables rerankers and language models to distinguish between functional descriptions and technical constraints. For example, a block entry would appear as: "Name: RS-485 Transceiver. Detailed Function: Supports full-duplex differential communication. Constraints: Operates up to 10 Mbps. Tags: interface, serial, communication."

These entries are then embedded into two representational spaces. Sparse representations are generated using SPLADE [53] (obtained from huggingface `naver/splade-cocondenser-ensembledistil`), which assigns learned importance weights to vocabulary terms based on masked language modeling. This produces a high-dimensional sparse vector capturing the relative importance of subword tokens. Dense representations are computed using a local instance of BGE-M3 [54], which encodes the same text into fixed-length float vectors via transformer pooling.

The indexed collection is hosted on a local Milvus [55] instance. Each document is stored with its `BlockID`, original text, SPLADE sparse vector, and BGE-M3 dense vector. SPLADE vectors are indexed using an inverted structure with inner product (IP) similarity, while dense vectors use a flat index, also with IP scoring. Milvus was chosen over alternatives (e.g., Facebook AI Similarity Search, Pinecone) due to its native support for sparse-dense hybrid search, compatibility with local CPU-only environments, and minimal setup overhead. The database is populated once and held in memory for subsequent queries.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

This resulting indexing process requires no manual tagging, synonym expansion, or template engineering, and it supports fully offline operation over a moderately sized corpus.

3.4.3 Requirement Querying and Scoring. Each requirement is treated as a standalone natural-language query. These often contain functional intentions expressed with varying levels of precision, from explicit protocol references, e.g., "JTAG" to implicit behavioral cues, e.g., "simultaneous two-way communication." To accommodate both, the system encodes each query into two parallel representations: a sparse lexical vector and a dense semantic embedding.

SPLADE converts the input into a weighted sparse vector over subword tokens. This allows context-aware expansion: latent terms not present in the original text (e.g., "duplex" and "bus") may be surfaced and weighted, enabling match against technical metadata without requiring exact lexical overlap. In contrast, the BGE-M3 [56] encoder maps the entire sentence to a fixed-length dense vector, capturing relational meaning across phrasings (e.g., "digital interface" versus "UART").

Both vectors are submitted to a local Milvus instance using separate similarity requests. Sparse vectors query an inverted index; dense vectors use flat approximate nearest neighbor search. Milvus fuses the scores using a built-in rank fusion strategy via *reciprocal rank fusion*, returning the top-k candidates (typically $k = 10$) with combined relevance scores.

An optional reranking stage refines these results. A cross-encoder reranker (BGE-Reranker) takes the raw requirement and each candidate's metadata string as joint input and produces a final scalar relevance score. Unlike precomputed embeddings, this model attends over full token-level interactions. It is applied only to the top-k shortlist, returning a re-ordered top-r set (e.g., $r = 5$). All models run locally on CPU, and no cloud dependencies are introduced.

The retrieval stage searches only `FunctionalName`, `DetailedFunction`, `Constraints`, and `SimilarityTags` fields and, without a minimum score cutoff, returns the circuit-block IDs of the top k hits.

3.4.4 Evaluation Setup and Metrics. To evaluate the system, we adopted an approach conceptually similar to recent benchmarking studies—such as the use of the pass@k/n metric in CIRCUIT [32], which measure model performance over structured design data. In our case, given the retrieval-based nature of the task, we used the *Hit@k* metric to assess how effectively relevant circuit blocks are retrieved in response to natural-language requirements. Each requirement in the dataset is associated with one or more relevant `BlockIDs`, which serve as ground truth labels. Given a query, the system returns a ranked list of candidate blocks. A prediction is considered correct if any of the annotated target blocks appears within the top-k results.

For example, if requirement R042 is annotated to map to block `T138`, and the retriever returns the ranked list `[T222, T045, T138, T317, T059]`, then Hit@5 is considered successful, since `T138` appears in the top 5. In scenarios where downstream components—such as LLM-based validators—consume only the highest-ranked result, the position of the correct block becomes critical. To capture this, we also report the *mean reciprocal rank* (MRR), which reflects how early the correct block appears in the list.

Evaluations are conducted over two subsets of the dataset. The 1-to-1 subset includes 271 requirements that each map to a circuit block. This subset supports precise computation of ranking metrics such as Hit@1, Hit@5, Hit@10, and MRR. The full set contains 328 requirements, including those with multiple correct blocks or no assigned match. This broader set provides insight into system robustness under ambiguity, distractor noise, and under-specification. We also logged the SPLADE similarity scores together with the IDs of the retrieved blocks. However, they are not directly interpretable as feature coefficients, as SPLADE uses learned token weights to compute relevance. Exploring token weight patterns to better understand how domain terms affect retrieval is left as future work.

To ensure consistent comparison across retrieval modes and isolate the contribution of each component to final performance, four retriever configurations are compared: sparse-only retrieval using SPLADE (1), hybrid retrieval with fused sparse and dense scores (2), and their cross-encoder reranking added variants (3–4). All queries should be processed using the same system settings, and no adaptation or feedback is applied during evaluation.

This evaluation is scoped to block-level matching in embedded control electronics using the fixed R3SET corpus (92 curated blocks and 100 synthetic distractors), under a CPU-only, offline setting. The reported results are intended as a reproducible reference baseline on R3SET for comparing retrieval and selection components, not as a general-purpose benchmark of design reuse across domains or full-system design tasks.

3.5 Generation for an Explainable Final Decision: Circuit Recommendation With a Rationale. To complement the retrieval pipeline with an interpretable output, we incorporate an LLM as a final decision layer, addressing *RQ3*. Details on model selection, prompt design, and evaluation are provided in the following subsections. This layer receives the top-k retrieved blocks along with the original requirement and produces both a ranked recommendation and a rationale. The aim is to emulate expert reasoning in selection and justification, supporting human validation and trust.

We adopt *Qwen2.5-72B*, a multilingual instruction-following open-weight LLM with a permissive license, in its 4-bit quantized form, hosted locally via llama.cpp for reproducibility and efficiency. It can deliver strong results on our domain-specific requirement classification task even under zero-shot prompts [27]. The selection was further guided by two considerations. First, at the time of writing, Qwen2.5 consistently ranks among the best-performing open-source models on instruction-tuning benchmarks such as MMLU (*Measuring Massive Multitask Language Understanding*) and GSM8K (*Grade School Math 8K*), especially in multilingual and complex reasoning tasks. Second, the quantized version balances performance and resource constraints, enabling local inference over 5–10 candidate blocks per query on CPU or a single graphics processing unit (GPU).

The model receives a structured zero-shot prompt (see Appendix A) including the requirement, top-k candidates with their metadata (function, constraints, and key components), and is asked to select the most relevant block and explain its reasoning.

Finally, we note that the generation-based decision layer is *optional*. In our pipeline, it operates downstream of hybrid retrieval and reranking, and is scoped strictly to the final decision task. Other generation-based applications (e.g., elicitation, constraint completion) are beyond the scope of this article and are discussed as future extensions.

3.5.1 Evaluation of Selection Accuracy. To evaluate the final decision layer, we first measure the model's top-1 selection accuracy against DE-designated labels for each requirement-block pair. Given that the matching scheme in our requirement set is predominantly one-to-one, we evaluate LLM outputs on a strict *match–no match* basis: model predictions are considered correct only if they exactly align with these labels, including accepted alternatives. To assess the contribution of internal reasoning of the LLM, we also report the highest-scoring output of the retrieval module as a reference. In addition, we also structure our evaluation across three key axes of variation to better understand the factors that influence model performance in circuit selection: requirement author, source project, and functional category. These dimensions were chosen to capture distinct sources of linguistic, contextual, and structural diversity present in real-world hardware design workflows:

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

(i) Requirement author dimension captures the impact of author-specific variation in language use, structure, and terminology. Since requirements may be authored by individuals with varying levels of technical background and expression styles, this breakdown helps assess the model's robustness to lexical diversity and surface-level formulation differences across otherwise similar functional intents.
(ii) Source project dimension examines cross-project generalization. Different source projects often differ in architectural conventions, documentation practices, and design constraints. Evaluating performance across source projects provides insight into the model's ability to adapt to domain-specific idioms and to handle shifts in project-level context or system design assumptions.
(iii) Functional category dimension refers to hardware-oriented operational groupings such as power conversion, control logic, and signal driving. Each category entails distinct engineering semantics, component behaviors, and interface expectations. By analyzing performance across these categories, we assess the model's capacity to recognize and reason over structurally and semantically distinct functional roles within hardware systems.

3.5.2 Evaluation of Generated Rationales. As a second evaluation axis at the final decision stage, we assess the quality of the LLM-generated *decision rationales* via a DE review, complementing retrieval-correctness metrics. We evaluate rationales produced by the best-performing offline LLM configuration, since this setting represents the most realistic deployment candidate under our low-resource constraints. To limit expert effort and time cost while preserving representativeness, we sampled approximately one-third of the full requirement-candidate set (total of 99 cases) using random stratified sampling to match the original categorical distributions (e.g., functional role, source project, and author). The evaluation guideline specifies the key quality dimensions; the full rubric definitions are provided in Appendix B.

The same DEs independently rated the stratified subset (i.e., 198 total ratings) using the evaluation guideline. For each case, experts scored six 1–5 Likert criteria—correctness, completeness, clarity, technical accuracy, usefulness, and trustworthiness—and annotated the rationale with a dominant `PrimaryPattern` label capturing the main quality/issue mode, optionally marking issue flags (e.g., hallucinated feature, missing key component, and requirement misunderstood). To characterize rating consistency in a way that reflects practical engineering use (where 4 versus 5 is typically acceptable but 5 versus 3 warrants review), we report tolerance-based agreement (% within ± 1 point) and the rate of material disagreements (rating gaps ≥ 2) per criterion.

The resulting data are used to characterize the rationales across two dimensions: quantitative quality scores and qualitative issue patterns. We summarize the six-criterion ratings by reporting, per criterion, n, mean $\pm$ SD, and the share of high (4–5) and low (1–2) scores across the 198 individual ratings. Inter-rater consistency is computed over the 99 cases as (i) tolerance agreement, defined as the percentage of cases with $|r_1 - r_2| \leq 1$ and (ii) material-disagreement rate, defined as the percentage with $|r_1 - r_2| \geq 2$; `PrimaryPattern` annotations are summarized as counts and percentages over cases.

This expert-based assessment aims to provide initial evidence of when generated rationales support reuse-oriented explainability and when they become incomplete or technically misleading; a larger-scale, multirater reliability study remains future work.

Table 4 Retrieval performance of sparse, hybrid, and reranked (RR) variants, $n = 271$ (out of 338, excluding NM = 57, 1 − M = 10)

Metric	Hybrid with RR	Hybrid without RR	SPLADE with RR	SPLADE without RR
Hit@1	0.71	0.58	0.69	0.57
Hit@5	0.92	0.87	0.88	0.84
Hit@10	0.93	0.93	0.91	0.91
MRR@5	0.78	0.69	0.76	0.68
MRR@10	0.78	0.70	0.76	0.69

3.6 A Pilot Study: Practical Utility of Retrieval-Driven Reuse. To address *RQ4*, we designed a task-based pilot study. The aim of this small-scale user-study is to obtain an initial indication of the practical utility of retrieval-driven reuse in realistic schematic design work.

Following a standardized pilot protocol, we defined two experimental conditions for each requirement: *FROM-SCRATCH* (FS), where the designer received only the textual requirement and designed the block using their normal workflow, and *REUSE-ASSISTED* (A), where the designer receives the same requirement plus the system's top-suggested candidate block and a short rationale explaining the suggestion. Designers were free to (i) reuse the suggestion as-is, (ii) adapt/modify it, or (iii) ignore it and proceed from scratch; if ignored, they could optionally inspect additional candidate blocks. The tasks were assigned to conditions in a counterbalanced design within the subject, so that comparable requirements appear in both conditions, reducing task-specific confounding [57] such as learning effect, ordering effects, or individual variation.

Despite the counterbalanced design, we introduced a *warm-up* task: the designers began the session with the same initial requirement as a short warm-up to familiarize themselves with the task format and time recording procedure. We decided not to collect or analyze data from this first warm-up design, and it is excluded from the reported task counts and descriptive comparisons.

We recruited two external junior electronics designers (approximately one year of professional experience; not involved in building R3SET or the retrieval system). Each designer completed nine schematic-block design tasks (18 total task instances). These tasks were randomly selected using numbers (1–366) generated by a true random number generator to represent requirement IDs. The tasks were limited to schematic capture only (no PCB layout/mechanical integration) and the designers were allowed to use their normal workflow and internet resources, but could not ask others for design help. Each task had an intended maximum duration of 30 min, and the designers recorded completion time per task.

For every task, the observer recorded task completion time and collected the resulting schematic. Immediately after each task, designers completed a Likert-style NASA-TLX [58] inspired five questions with 5-point ratings of perceived task difficulty, time pressure, and confidence in the final design; for REUSE-ASSISTED tasks, they additionally rated the perceived helpfulness of the suggested blocks. A DE then rated each design on a three-point correctness scale (0 = major issues, 1 = minor fixable issues, 2 = fully acceptable). We summarize the outcomes using descriptive comparisons between conditions (time, correctness, and subjective ratings) rather than formal hypothesis testing, reflecting the small sample size and illustrative intent.

4 Experimental Results and Discussion

This section presents the empirical evaluation of our retrieval and selection pipeline, structured to address research questions. We assess retrieval performance (Sec. 4.1) and robustness (Sec. 4.2) addressing design decisions; examine the behavior of the LLM-based selector on identifying the correct reusable circuit block among retrieved candidates (Sec. 4.3); discuss qualitative aspects of the dataset (Sec. 4.4); and assess the practical reuse utility of the requirements-driven retrieval (Sec. 4.5). All results are reported on the introduced R3SET dataset.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Table 5 Hit@k and MRR distributions for retrieved blocks for non-NM requirements via hybrid (SPLADE + BGE-M3) retrieval, and with BGE reranking $n = 271$ (out of 338, excluding NM = 57, 1 − M = 10)

		Hit@1	Hit@5	Hit@10	MRR@5	MRR@10	n
Functional category	C	0.77	0.96	0.96	0.83	0.83	26
	D	0.74	0.87	0.87	0.78	0.78	46
	F	0.79	0.97	0.99	0.86	0.86	67
	I	0.65	0.92	0.93	0.75	0.75	75
	P	0.63	0.86	0.89	0.70	0.70	57
Source project	P001	0.81	1.00	1.00	0.88	0.88	36
	P002	0.74	0.97	0.97	0.82	0.82	38
	P003	0.66	0.90	0.90	0.73	0.73	29
	P004	0.70	0.90	0.90	0.77	0.77	20
	P005	0.75	0.88	0.88	0.79	0.79	40
	P006	0.73	0.94	0.94	0.81	0.81	49
	P007	0.67	0.91	0.94	0.74	0.75	33
	P008	0.54	0.77	0.88	0.63	0.64	26
Requirement author	DE1	0.63	0.88	0.91	0.72	0.72	82
	DE2	0.72	0.94	0.94	0.79	0.79	88
	DE3	0.76	0.92	0.93	0.82	0.82	101
Overall		0.71	0.92	0.93	0.78	0.78	271

4.1 Retrieval Performance: Sparse, Hybrid, and Reranked Variants. Table 4 compares the performance of four configurations: SPLADE-only, hybrid (SPLADE + BGE-M3 dense retrieval), and versions with or without BGE reranking. The results are reported on a filtered set of 271 requirements with one-to-one matches. The cases involving no-match or compound relations are excluded from this analysis to ensure strict upper-bound comparability.

Across all metrics, the hybrid system with reranking achieves the strongest performance, reaching 71% Hit@1 and 93% Hit@10, along with the highest MRR scores.

The studies in the literature suggest [59,60] that the reranker can mask retrieval weaknesses and may misinterpret specialized jargon in domains such as electronics. However, in our case, adding the reranker consistently improves retrieval, particularly at top ranks (Hit@1), aligning with Ref. [61]. Without reranking, performance drops sharply, confirming the benefit of semantic reranking for disambiguating near-matches. SPLADE alone performs competitively, but the hybrid model achieves better overall recall.

Our retrieval pipeline delivers strong Hit@10 on the curated circuit-block library, but its effectiveness is conditioned by several design choices. The index text is a concatenation of FunctionalName, DetailedFunction, Constraints, and SimilarityTags; this deterministic construction aligns closely with our requirement phrasing and likely amplifies lexical overlap, especially through Tags. Because we run SPLADE and the optional cross-encoder strictly in inference mode, classical overfitting is unlikely; however, measured gains may be inflated relative to requirements used for evaluation. To address this, systematic procedures are adopted to extract them: the requirements were authored by three experienced domain experts, and we preserved original wording, grammar, typos, and style; we did not normalize units, symbols, or phrasing. This strengthens external validity compared to sanitized benchmarks and ensures SPLADE and the reranker face naturally occurring jargon, shorthand (e.g., 3V3 versus 3.3 V), and idiosyncratic constructions.

4.2 Retrieval Robustness. To assess how retrieval performance generalizes (RQ2) across requirement authors, functional domains, and source projects, we report stratified metrics in Table 5. The performance is reported using hit rate and MRR values for $k = 1$, 5, and 10, along with the corresponding sample size (n) for each category.

Overall Hit@10 and Hit@5 remain high and closely aligned (93% and 92%, respectively), indicating that most relevant candidates are consistently retrieved within the top-5 results. The small gap suggests that Hit@5 provides a sufficient and practically appropriate operating point for the selection stage, where the goal is to surface a small, high-quality candidate set rather than enforce a single best prediction. In contrast, Hit@1 varies substantially:

- By *requirement author*, Hit@1 ranges from 63% (DE1) to 76% (DE3)
- By *functional category*, feedback (F) achieves 79%, while power (P) drops to 63%
- By *project*, P001 shows the highest Hit@1 (81%), while P008 is lowest (54%)

This variation suggests that retrieval precision might be sensitive to expression style and domain-specific terminology. Author-dependent patterns, such as preferred phrasing or unit conventions, influence lexical matching. Likewise, blocks in structurally compact domains (e.g., feedback) appear easier to retrieve more reliably than blocks in ambiguous or interface-heavy domains. Because author, project, and functional category can be partially correlated and requirement difficulty can vary across subsets, we interpret these subgroup results descriptively rather than as isolated factor effect.

These results confirm the pipeline's ability to return relevant candidates under realistic variation, and also point to limitations in capturing underspecified or compositionally complex requirements. For example, how requirements are formulated, with certain expression styles or content types aligning more naturally with the retrieval mechanism. This implies a need for adaptation to linguistic or structural patterns specific to different domains in order to improve precision in early ranks.

Beyond domain-specific language patterns, retrieval performance is also shaped by systemic factors in both corpus construction and model behavior. For example, our small expert pool may exhibit shared habits that align with indexed fields particularly SimilarityTags, leading to inflated lexical overlap. As a result, performance may degrade in cases involving novice-written specs, rigid vendor templates, multilingual variation, or differing organizational styles. Additional caveats stem from technical artifacts such as domain glyph tokenization (e.g., I^2C versus I2C, μ-ohm symbols, 3V3 versus 3.3 V) and potential truncation in *VARCHAR* fields. Moreover, several aspects of the retrieval pipeline—uniform field concatenation (i.e., no field-specific weighting), reliance on sparse-only retrieval (limiting paraphrase and numeric equivalence), and tokenizer quirks—can all influence outcomes, independent of authorship effects. Taken together, these factors, not merely the cleanliness of input queries, represent key considerations when interpreting hit rate metrics.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

4.3 Large Language Model-Guided Decision Behavior. We evaluated the behavior of the final decision layer along two axes: the model's top-1 selection accuracy against the DE-designated labels for each requirement-block pair and expert review of a stratified subset of generated decision rationales. Unlike retrieval performance and robustness evaluation, we report these results by including NM requirements because the LLM decision layer must be evaluated not only on selecting the correct block when a match exists but also on its tendency to produce false-positive selections when no suitable block is available.

4.3.1 Large Language Model's Performance in Identifying the Correct Circuit. Table 6 presents the LLM's performance in identifying the correct circuit block among the top five-retrieved candidates, addressing *RQ3*. The comparison includes one baseline configuration using only the retriever (ExpM), and three configurations that combine the retriever with the Qwen2.5-72B model under varying prompt designs and quantization levels: the baseline (ExpM) uses only the top-ranked result from the retriever without LLM assistance. The ExpQ02 variant adds Qwen2.5-72B (Q6 quantization) with a persona-style instruction. ExpQ04 uses a lighter Q4 quantized version of the same model with both persona prompting and chain-of-thought (CoT) reasoning (see Appendix A). ExpQ03 mirrors ExpQ04's prompt strategy but with Q6 quantization. In addition, while utilizing open-source models addresses the industrial needs, ensures reproducibility and auditability, it limits the use of higher-performing closed models. To address this, we also included four commercial reference configurations in Table 6, where the same prompt from ExpQ03 and ExpQ04 is used with a range of model variants from OpenAI: GPT-5.2 (ExpG01-H2), GPT-5.1 (ExpG01-H), GPT-5-mini (ExpG01-M), and GPT-5-nano (ExpG01-N). These closed-source and online models violate our constraints but indicate what state-of-the-art commercial LLMs can achieve in the same decision role.

In addition to 1:1 and NM cases, we excluded the identified 1:M requirements (10 in total) where satisfying the requirement requires composing multiple blocks simultaneously. These cases were explicitly annotated in R3SET, but their limited count does not support reliable or generalizable quantitative analysis, so we excluded them from the main retrieval evaluation. We expect that handling such composite reuse scenarios will require an additional input-elicitation or "active retrieval" layer that can decompose a requirement into functional subproblems and retrieve a coordinated set of blocks across functional categories. Designing and validating such a mechanism is beyond the scope of the present study and remains an important direction for future work.

When evaluated on the full 328-requirement set (excluding only the 10 one-to-many cases), the retrieval-only baseline achieves 58.5% decision accuracy at Hit@1. At first glance, this appears to be a substantial drop from the 71% reported earlier; however, this difference is attributable to the metric itself: standard Hit@k metrics necessarily treat all NM queries as errors, since no correct retrieval exists by definition. This behavior is therefore expected and, in practice, confirms the need for an additional no-match detection layer that can explicitly recognize NM requirements in realistic deployment settings.

To address the concern that subgroup differences by requirement author, source project, and functional category may be confounded (e.g., certain authors primarily working on specific projects), we further performed a multifactor analysis that estimates method effects while jointly controlling for these factors. Specifically, we fit a binomial logistic generalized estimating equation (GEE) model [62] clustered by requirement (RID), with method as the primary factor and author/project/category as covariates (noncomposite ground truth only, $n = 328$). Table 7 summarizes the adjusted comparisons using `ExpQ03` as the reference configuration. After adjustment, the overall ranking is unchanged: the `ExpG01-*` configurations yield significantly higher odds of correct block selection than `ExpQ03` (OR $\approx$ 1.39–1.50, $p \leq 0.02$), whereas `ExpQ02` underperforms (OR ≈ 0.82, $p \approx 0.018$) and `ExpQ04` is statistically indistinguishable from `ExpQ03`. This indicates that the method-level differences observed in Table 6 are not attributable solely to correlations among subgroup factors.

The highest-performing offline configuration (ExpQ03) achieves 60.1% decision accuracy on the complete dataset, slightly above the retriever-only baseline. As a common reference point, all commercial cloud-based alternatives further increase accuracy up-to 68.9% offering incremental accuracy gains in selecting among top five-retrieved candidates. Three patterns emerge:

- *Prompting matters*: Persona pattern without CoT underperforms across all dimensions, confirming its effect on the model's reasoning.
- *Quantization tradeoffs exist*: Lower-precision variants (Q4) do not consistently degrade performance, but subtle effects on accuracy are observed on certain application domains and functional categories.
- *False positives are common*: The LLM correctly identifies NM cases without being influenced by distractors in only 15 out of 57 instances.

These findings indicate that while the LLM can select correctly in many cases, its accuracy is bounded by two factors: (1) the quality of the requirement and candidate metadata and (2) the

Table 6 Circuit-block selection accuracy with LLMs by category $n = 328$ (out of 338, excluding 1−M = 10)

Section	Group	ExpM	ExpQ02	ExpQ04	ExpQ03	ExpG01-H2	ExpG01-H	ExpG01-M	ExpG01-N	n
Functional category	C	64.5%	38.7%	51.6%	45.2%	48.4%	61.3%	71.0%	54.8%	31
	D	63.0%	55.6%	61.1%	53.7%	61.1%	63.0%	64.8%	74.1%	54
	F	68.8%	76.6%	77.9%	79.2%	80.5%	74.0%	79.2%	75.3%	77
	I	52.1%	43.6%	53.2%	52.1%	70.2%	66.0%	58.5%	60.6%	94
	P	50.0%	55.6%	51.4%	61.1%	69.4%	70.8%	72.2%	68.1%	72
Source project	P001	80.6%	41.7%	63.9%	63.9%	77.8%	61.1%	80.6%	75.0%	36
	P002	68.3%	53.7%	56.1%	58.5%	70.7%	78.0%	82.9%	70.7%	41
	P003	52.8%	63.9%	58.3%	52.8%	69.4%	75.0%	69.4%	69.4%	36
	P004	40.0%	40.0%	48.6%	51.4%	57.1%	60.0%	60.0%	65.7%	35
	P005	62.5%	54.2%	62.5%	60.4%	70.8%	58.3%	64.6%	68.8%	48
	P006	61.0%	61.0%	62.7%	62.7%	64.4%	66.1%	64.4%	59.3%	59
	P007	51.2%	58.1%	58.1%	62.8%	74.4%	74.4%	65.1%	69.8%	43
	P008	46.7%	70.0%	66.7%	66.7%	66.7%	73.3%	63.3%	63.3%	30
Requirement author	DE1	52.0%	49.0%	58.0%	58.0%	65.0%	59.0%	67.0%	67.0%	100
	DE2	59.4%	58.5%	60.4%	64.2%	68.9%	69.8%	65.1%	64.2%	106
	DE3	63.1%	58.2%	60.7%	58.2%	72.1%	73.8%	73.0%	70.5%	122
Overall		58.5%	55.5%	59.8%	60.1%	68.9%	68.0%	68.6%	67.4%	328

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Table 7 Multifactor adjusted comparison of circuit-block selection accuracy against the primary configuration (ExpQ03, reference)

Method	Acc. (%)	ΔAcc. (pp)	Adj. OR (95% CI)	*p*
ExpQ03	60.1	0.0	Ref.	–
ExpQ04	59.8	−0.3	0.99 [0.85, 1.15]	0.862
ExpM	58.5	−1.5	0.94 [0.72, 1.21]	0.615
ExpQ02	55.5	−4.6	0.82 [0.70, 0.97]	0.018
ExpG01-N	67.4	+7.3	1.39 [1.10, 1.77]	0.007
ExpG01-H	68.0	+7.9	1.43 [1.10, 1.86]	0.007
ExpG01-M	68.6	+8.5	1.48 [1.15, 1.89]	0.002
ExpG01-H2	68.9	+8.8	1.50 [1.16, 1.94]	0.002

Notes: Adjusted odds ratios (ORs) from a multifactor logistic GEE clustered by requirement (RID), with covariates for requirement author, source project, and functional category; noncomposite ground truth requirements only ($n = 328$). Acc. is overall selection accuracy; ΔAcc. is absolute difference versus ExpQ03. OR > 1 indicates higher odds of correct selection than ExpQ03.

model's limited capacity for numerical reasoning and logical composition. Notably, the LLM often selects blocks that are *plausible* but incorrect, particularly when requirements are vague or span multiple functions. However, the models never *hallucinate* new circuits. They always select from the curated library of existing blocks. The failure mode in the NM cases is not fabrication of nonexistent hardware, it always picks from the existing library. In no-match cases, the failure is accepting "near misses"—blocks with similar functions but wrong specifications (e.g., 3.3 V sensor when 5 V is required). This implies that the model cannot reliably reject these constraint violations on adversarial test cases. In practice, this motivates using the LLM's output as a *triage aid* under expert supervision, rather than as an autonomous reuse selector. Therefore, the primary value of this experiment should be considered as diagnostic rather than incremental performance. It makes visible when and how a constrained LLM misjudges retrieved blocks (especially in NM and underspecified cases) and highlights the risks of delegating final block selection to such a model without expert oversight.

These findings suggest that the final decision step remains a bottleneck, particularly in resolving underspecified or multitarget inputs. Nonetheless, the results confirm that the hybrid pipeline reliably retrieves and ranks relevant blocks, while highlighting areas where LLM-guided selection should be improved.

4.3.2 Large Language Model's Capacity for Decision Explainability. In alignment with *RQ3*, we evaluate whether the LLM-generated *decision rationales* provide actionable explainability for reuse-oriented workflows. Table 8 summarizes the outcomes over aggregated 198 individual DE ratings. Overall, rationales were rated positively (descriptive index averaged over Q1–Q6: 4.14 ± 1.07). Clarity was the strongest dimension (4.51 ± 0.91), with 88.9% of ratings ≥4, indicating that explanations are typically readable and well structured. In contrast, technical accuracy was the weakest dimension (3.93 ± 1.25) and exhibited a heavier low-score tail (14.7% of ratings ≤2). This suggests that the main limitation is not presentation quality but occasional technically incorrect statements, which can also depress downstream trustworthiness (4.03 ± 1.35; 17.7% ≤2) and usefulness (4.04 ± 1.33; 16.7% ≤2) despite high clarity.

The qualitative coding (Table 8, panel B) shows that 71.7% of expert annotations labeled the rationale as GOOD_RATIONALE, while the most frequent failure modes were REQ_MISUNDERSTOOD (9.6%) and KEY_COMPONENT_MISSING (6.1%). In addition, while the model generates valid block recommendations, it occasionally attributes nonexistent properties to the block when justifying its choice. Although this specific error type, HALLUCINATED_FEATURE, was less common (3.5%), it remains consequential because fabricating technical capabilities can disproportionately harm trust in engineering decision support.

Inter-rater consistency was characterized over per case ($n = 99$), using tolerance-based agreement to reflect practical use (where 4 versus 5 is typically acceptable, but larger gaps merit inspection). We observed a systematic severity offset: DE2 tended to score

Table 8 Expert review of generated rationales

A. Likert criteria (1–5), expert ratings

Criterion	*n*	Mean ± SD	%≥4 (4–5)	%≤2 (1–2)
Correctness (Q1)	198	4.28 ± 1.19	80.3	11.1
Completeness (Q2)	198	4.06 ± 1.09	78.8	10.6
Clarity (Q3)	198	4.51 ± 0.91	88.9	4.0
Tech. accuracy (Q4)	198	3.93 ± 1.25	72.2	14.7
Usefulness (Q5)	198	4.04 ± 1.33	74.2	16.7
Trustworthiness (Q6)	198	4.03 ± 1.35	73.7	17.7
Overall index (avg Q1–Q6)	198	4.14 ± 1.07	71.2	7.6

B. Dominant PrimaryPattern (expert annotations)

PrimaryPattern	*n*	%
GOOD_RATIONALE	142	71.7
REQ_MISUNDERSTOOD	19	9.6
KEY_COMPONENT_MISSING	12	6.1
MIS_CITED_FEATURE	9	4.5
HALLUCINATED_FEATURE	7	3.5
POSTHOC_RATIONALIZATION	6	3.0
TOO_VAGUE	3	1.5

Note: Panel A aggregates individual DE Likert ratings (1–5) over 99 cases × 2 experts ($n = 198$ ratings). Panel B reports the distribution of dominant PrimaryPattern labels over the same expert annotations ($n = 198$).

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Table 9 Pilot study outcomes comparing schematic-block design FROM-SCRATCH versus REUSE-ASSISTED, excluding the warm-up task

Metric	FROM-SCRATCH ($n = 8$)	REUSE-ASSISTED ($n = 8$)
Time (min), median (IQR)	27.0 (15.8–30.0)	21.0 (17.8–23.5)
Time (min), mean ± SD	27.25 ± 15.00	20.25 ± 7.85
Correctness mean (0–2)	0.75	1.00
Correctness distribution [0/1/2]	2/6/0	3/2/3
Clarity (Q1, 1–5), mean	2.88	4.00
Assumptions (Q2, 1–5), mean	2.63	1.75
Difficulty (Q3, 1–5), mean	2.75	2.50
Time pressure (Q4, 1–5), mean	3.00	2.75
Confidence (Q5, 1–5), mean	2.50	3.25
Helpfulness (Q6, A only, 1–5), mean	–	4.25

higher than DE1 by approximately 0.33 to 0.51 points on most criteria (Q1, Q2, Q4–Q6), while clarity was nearly aligned (DE2 lower by ≈0.05). Given this offset and the strong ceiling skew toward high ratings (many 4–5 scores), we treat raw/tolerance agreement as the primary reliability indicator. Across Q1–Q6, experts were within ±1 point on 71.7–88.9% of cases (highest for clarity), while material disagreements (rating gaps ≥2) occurred in 11.1–28.3% of cases depending on the criterion. For `PrimaryPattern`, experts matched exactly on 63.6% of cases.

4.4 Qualitative Assessment of the Dataset. *RQ1* concerns the design of a dataset suitable for developing and evaluating requirement-driven retrieval in data-scarce, heterogeneous domains. The literature considers the diversity and internal consistency of hardware-related datasets important [26]. The requirements corpus was initially constructed for functional classification, but its structure, authorship, and diversity rendered it adaptable for retrieval evaluation. In our approach, we address this challenge by proposing an integrated framework that combines both expert insight and automated support, thereby reducing manual overhead while preserving semantic richness. As for consistency, it must be ensured at multiple levels: between the extracted features and their source data, across different data modalities, and within expert-provided annotations. To this end, we adopt a dual strategy that combines LLM-as-a-judge validation mechanisms with domain-expert supervision guided by structured annotation guidelines.

Empirically, retrieval experiments confirmed that the dataset supports realistic testing. Requirements vary in phrasing, abstraction, and formality, reflecting how hardware specifications are written in practice. For example, minor inconsistencies such as "3V3" versus "3.3" or ambiguous quantifiers like "high-resolution" contributed to retrieval mismatches that are useful for evaluating system robustness. Likewise, NM annotations helped identify system-level or COTS-referenced requirements outside the block library's scope.

Structurally, the dataset includes one-to-one, one-to-many, and NM mappings, enabling coverage of common real-world retrieval outcomes. The requirements span five functional categories and eight distinct hardware projects, ensuring cross-domain generalization. Authorship diversity introduces natural lexical and syntactic variation, as evidenced by performance differences among the authors in Sec. 4.2.

However, annotation process and the experiment errors revealed that the corpus also exposes inherent limitations of real-world documentation. Some requirements lack hardware-relevant constraints or express vague design intents. With the exception of 5 of 338 requirements for which we corrected an obvious single-token typographical/value error (e.g., a mismatched input/output range or an incorrect protocol keyword) to avoid deterministic label noise that would otherwise eliminate discriminative instances in a modest-sized corpus, these defects were not artificially corrected, as their presence enables meaningful assessment of retrieval performance under imperfect conditions. That said, we came to view these issues not as dataset limitations per se, but rather as reflections of the inherent challenges found in real-world, natural-language requirements. Given that requirements documents are living artifacts subject to ongoing revision and interpretation, such inconsistencies are, in our view, part of the natural lifecycle of the specification process. This insight led us to identify a promising direction for future work: incorporating a dedicated information elicitation step similar to Ref. [63] prior to feeding requirements directly into retrieval systems.

4.5 Practical Assessment of Requirement Retrieved Blocks for Reuse. *RQ4* assesses whether requirements-driven retrieval provides practical reuse utility, i.e., reducing design time and perceived effort without degrading correctness. Table 9 summarizes the pilot outcomes (excluding the warm-up task `T00`). Across the remaining tasks REUSE-ASSISTED (A: $n = 8$), reduced completion time (median 21 min, IQR (interquartile range) 17.8–23.5) compared to FS ($n = 8$), (median 27 min, IQR 15.8–30.0), while maintaining or improving expert-rated correctness (mean 1.00 versus 0.75; fully acceptable designs: 3/8 in A versus 0/8 in FS). Subjective ratings were also favorable to assistance: designers reported higher requirement clarity (4.00 versus 2.88), fewer missing-detail assumptions (1.75 versus 2.63), and higher confidence (3.25 versus 2.50); suggested blocks were rated as helpful on average (4.25/5). Given the small sample, we report descriptive comparisons; note that the intended 30-min cap was not strictly met in two instances (FS: 60 min; A: 33 min).

Based on designers' post-task interview (see Appendix C) responses, our qualitative observations suggest that candidate blocks were most useful for quickly establishing an initial circuit concept and for sanity-checking uncertain designs. Conversely, designers typically ignored the suggestions when the requirement was perceived as repetitive, when they already had a clear solution in mind, or when the retrieved block appeared mismatched to the requirement.

5 Limitations and Future Research Directions

The proposed framework provides a reproducible baseline for requirement-driven reuse, but its scope is shaped by several practical and methodological constraints. These factors limit generalization in some areas while pointing to directions where targeted extensions could strengthen system robustness and applicability.

First, the dataset is limited to eight projects, selected for open licensing and functional diversity. While this introduces variation in authoring style and hardware types, it may not capture the full spectrum of schematic practices. Expansion to more domains and contributors is needed to reduce stylistic coupling and improve generalizability. In addition, our approach may introduce lexical coupling between requirements and metadata particularly through fields like SimilarityTags and DetailedFunction. We mitigated this by preserving original writing styles, avoiding language normalization, and introducing 100 synthetic distractors with broader lexical

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

variation. Still, future work may benefit from evaluating generalization on independently authored corpora. To facilitate this, we release a replication package to enable community-driven expansion, broadening domain coverage and reducing stylistic/lexical coupling.

Second, the SPLADE-based sparse retriever assumes high lexical overlap, making it vulnerable to vague phrasing or semantic shifts. To reduce this sensitivity, the dataset preserves natural expert phrasing, including domain-specific shorthand and unit variants. Future systems may benefit from field-aware embeddings and indexing strategies.

Third, although the LLM-based selector provides negligible net gains in top-1 selection accuracy (58.5% and 60.1%) under our chosen local, quantized model, our results indicate that, in this configuration, the LLM should not be viewed as an accuracy-enhancing reranker but as an exploratory tool for generating rationales and revealing failure modes. Any deployment as an automated selector would require LLM-assisted requirement clarification before retrieval, which is a recommended future step, or domain-adapted/advanced models, which are outside the scope of this study.

Another key limitation is the current exclusion of one-to-many, many-to-one, and many-to-many requirement-block mappings. Realistic designs often involve compositions, e.g., a driver paired with a current sensor, or a shared power stage serving multiple subsystems. Our current annotation schema treats the latter two conservatively, labeling them as no-match or unmodeled. A future extension will introduce compositional labeling and multiblock retrieval capabilities.

Our calibration of the LLM-as-judge is also limited to a small in-domain subset (20 blocks) and a single expert group; while it provides a practical sanity check, a more systematic inter-rater reliability study is left for future work.

The exploratory pilot study of the practical assessment of reusability has several important limitations that constrain the interpretation of its findings. First, the small sample size ($n = 2$) precludes statistical inference and limits generalizability. The within-subject design introduces the possibility of carryover and learning effects, while the imposed time constraints may not accurately reflect real-world workplace practices. Because designers could ignore suggestions and optionally inspect additional candidates, the pilot evaluates practical assistance behavior (accept/modify/reject), not forced reuse. In addition, outcomes were evaluated by a single, non-blinded expert rater, introducing potential assessment bias. Accordingly, the findings should be interpreted as a proof-of-concept demonstrating the feasibility of integrating requirement-driven retrieval into schematic design workflows and its potential for time savings and perceived utility among junior designers. Robust validation of these preliminary patterns will require a larger-scale study (e.g., $n \geq 12$) employing a between-subject design, blinded evaluation, and participants with more diverse levels of expertise, which constitutes a key direction for future work.

Finally, the manual extraction of circuit blocks ensures ground-truth fidelity but limits scale. We partially address this by releasing our full metadata set and curation guidelines for benchmarking. Future work should explore the use of vision-language models for automated schematic segmentation.

In sum, while the pipeline is designed for transparency, reproducibility, and low-resource environments, broader adoption will require improvements in generalization, reasoning, and automated scaling. This work lays the foundation for such advances.

6 Conclusion

Design reuse in electronic hardware remains hindered by unstructured documentation, heterogeneous artifacts, and limited access to annotated datasets. This study addresses these challenges with a reproducible, low-resource-compatible approach that models the reuse process from natural-language requirement input to final circuit-block recommendation. By integrating expert-curated metadata, hybrid sparse-semantic retrieval, and LLM-based selection, we demonstrate a complete workflow suited for environments with restricted data, compute, or proprietary tool access.

The first contribution is an expert-guided metadata modeling framework that formalizes how design knowledge can be extracted, labeled, and validated using MLLMs under human supervision. This process ensures traceable and consistent metadata representations that support reliable retrieval.

The second contribution is R3SET, the first open-access dataset linking requirements to reusable circuit blocks, enabling principled evaluation of retrieval and reasoning methods under realistic and noisy conditions.

The third contribution is a hybrid sparse-semantic retrieval framework, which systematically combines SPLADE-based lexical expansion with BGE-M3 semantic embeddings under a unified scoring model. This design maintains interpretability for functional terms while expanding semantic coverage in underspecified queries, offering a practical path for low-resource environments.

Empirical evaluation shows that sparse lexical signals remain essential for maintaining technical traceability, while hybrid sparse-semantic fusion improves robustness against linguistic variation and domain phrasing. Sparse retrieval performs particularly well when requirement wording preserves domain-specific terminology, whereas performance declines for vague, underspecified, or multi-block queries. Although the LLM-based selector provides interpretable rationales, it often fails to demonstrate a strong accuracy boost relative to similarity-based ranking, indicating that reasoning over circuit metadata remains a challenge for quantized open-weight LLMs. Many observed errors arise from both model limitations and ambiguities inherent in the natural-language inputs. These findings suggest that retrieval effectiveness depends not only on the model but equally on the structure, precision, and completeness of both the requirements and the underlying metadata.

Overall, this work provides a complete, reproducible workflow and an openly shared dataset that together form a methodological foundation for future research in requirement-driven design reuse. The approach also offers a methodological basis that may be adapted to other engineering contexts where design artifacts are heterogeneous, resources are limited, and interpretable retrieval is required.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

To promote transparency, reproducibility, and community-driven expansion, we provide our code and dataset as a complete replication package through the Open Science Framework repository.[2]

Author Contribution Statement

E.B. Uyar led the conceptualization and methodology, developed and implemented the software, carried out the investigation, data curation, and formal analysis, prepared the visualizations, and took primary responsibility for writing—original draft, and writing—review and editing. A.E. Gursoy and C. Gokce equally contributed domain expertise to the methodology, participated in validation and data curation, and contributed to writing—review. T.T. Temizel provided supervision, contributed to methodology, and participated in writing—review and editing. All authors approved the final version of the article.

Appendix A: Prompt Templates

[2]https://osf.io/ucebp/

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Image-Metadata Extraction Prompt

```
You are an AI assistant specialized in hardware design. ←↩
      →Your task is to analyze circuit schematic block ←↩
      →images and prepare metadata according to the ←↩
      →guideline that provided. In the following prompts←↩
      →, I will share extracted block images, and I ←↩
      →expect you to generate metadata from these images←↩
      → in the required format. The image file names I ←↩
      →provide will be in the Txxx format, which differs←↩
      → from the format mentioned in the guideline. ←↩
      →According to the guideline, you will apply only ←↩
      →these deviations: when creating the metadata ←↩
      →table, write the image file name I provide in ←↩
      →both the BlockID and ProjectCode fields. For the ←↩
      →OriginalCoords field, write NA, and for the ←↩
      →Author field, write GPT. If there are any points ←↩
      →you would like to clarify before starting, you ←↩
      →may ask.
```

Synthetic Data Judge Prompt

```
You are an expert hardware design reviewer evaluating ←↩
      →structured metadata entries that describe ←↩
      →electronic circuit blocks. Each entry consists of←↩
      → multiple fields (e.g., FunctionalName, ←↩
      →DetailedFunction, KeyComponents, Constraints, etc←↩
      →.) written to support reuse in requirement-driven←↩
      → retrieval systems.

Your task is to:
1- Evaluate the semantic quality, field integrity, and ←↩
      →reuse readiness of each metadata row.
2- Use the attached Metadata Authoring Guidelines to ←↩
      →check for structural and stylistic conformance.
3- Identify concrete issues (e.g., vague descriptions, ←↩
      →misuse of trade names, missing constraints, ←↩
      →misaligned function tags).
4- For each issue, propose targeted corrections (field-←↩
      →level edits or rewrite suggestions).

Return for each metadata row:
 A binary judgment: ACCEPTABLE / NEEDS IMPROVEMENT
 A brief explanation (max 3 sentences)
 Field-specific correction suggestions (if any)
```

Zero-Shot Prompt w/o CoT

```
You are a circuit analysis assistant.
From the five circuit descriptions below, which one meets←↩
      → this hardware requirement: "{Requirement}"?
CIRCUITS:
{context}

Answer using the format below, if none of the circuits ←↩
      →satisfy the requirement, return 'NM' instead of ←↩
      →an ID.
<ANS>
Circuit ID: <ID or NM>
RATIONALE: <Max 12 words - highlight relevant match or ←↩
      →mismatch>
</ANS>
```

Zero-Shot Prompt with CoT

```
You are a circuit analysis assistant.
From the five circuit descriptions below, which one meets←↩
      → this hardware requirement: "{Requirement}"?
CIRCUITS:
{context}

Think step-by-step.
Answer using the format below, if none of the circuits ←↩
      →satisfy the requirement, return 'NM' instead of ←↩
      →an ID.
<ANS>
Circuit ID: <ID or NM>
RATIONALE: <Max 12 words - highlight relevant match or ←↩
      →mismatch>
</ANS>
```

Appendix B: Rationale Evaluation Definitions

Likert Questions. Q1_Correctness_1to5
Does the rationale clearly support the actually selected block (even if the selection is wrong)?
1 = clearly points to a wrong or mismatched block; 5 = clearly supports the selected block.

Q2_Completeness_1to5
Does the rationale include the key factors that a competent engineer would mention?
1 = misses important reasons; 5 = covers all critical decision points.

Q3_Clarity_1to5
Is the rationale easy to read and understand?
1 = confusing or hard to parse; 5 = clear, concise, and readable.

Q4_TechnicalAccuracy_1to5
Are the technical statements about the block and requirement correct?
1 = contains clear technical errors; 5 = no technical errors detected.

Q5_Usefulness_1to5
Would this rationale actually help you validate or challenge the decision?
1 = not useful; 5 = very helpful and actionable.

Q6_Trustworthiness_1to5
Based on this rationale alone, how much would you trust the system's recommendation?
1 = would not trust this system; 5 = would trust it for similar decisions.

Error Patterns and Flags. Exactly one PrimaryPattern value for each row was selected, representing the dominant quality or issue of that rationale.

- *GOOD_RATIONALE*—Grounded in the requirement and block, technically correct, and helpful.
- *HALLUCINATED_FEATURE*—Mentions features that the block does not have or that are not in the data.
- *TOO_VAGUE*—Generic wording; could apply to almost any block; lacks specifics.
- *MIS_CITED_FEATURE*—Uses real features but misattributes or confuses them (e.g., wrong direction, swapped roles).
- *REQ_MISUNDERSTOOD*—Shows a clear misunderstanding of the requirement (e.g., wrong range or interface).
- *KEY_COMPONENT_MISSING*—Ignores a critical component or constraint that should appear in a proper explanation.
- *POSTHOC_RATIONALIZATION*—Plausible story that clearly does not match the actual reasons (for example, refers to features that are irrelevant or that the system did not use).
- *OTHER*—Any case not covered above (briefly explained in comments).

Appendix C: Pilot Study Subjective Measures

Task Rating Form—Subjective Measures. For each task, designers rated the following items on a 1–5 Likert scale (1 = low, 5 = high).

TR1_RequirementClarity_1to5
Perceived clarity of the requirement text.
1 = very unclear; 5 = very clear.

TR2_AssumptionsMissingDetails_1to5
Extent to which missing details required assumptions or inference.
1 = none or almost none; 5 = many/major gaps.

TR3_TaskDifficulty_1to5
Overall perceived difficulty of the design task.
1 = very easy; 5 = very difficult.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

TR4_TimePressure_1to5
Subjective feeling of time pressure while working on the task.
1 = none or almost none; 5 = very high.

TR5_DesignConfidence_1to5
Confidence that the final design satisfies the requirement.
1 = not confident at all; 5 = completely confident.

TR6_CandidateHelpfulness_1to5
(REUSE-ASSISTED tasks only) Perceived usefulness of suggested candidate block(s).
1 = not helpful at all; 5 = extremely helpful.

Interview Form—Semistructured Questions. After completing all tasks, designers participated in a short, semistructured interview (or provided written responses) based on the prompts below.

- When did the candidate blocks help most?
- When were the candidate blocks ignored, and why?
- Did any suggested blocks mislead or confuse you? If so, how?
- For which kinds of tasks would you want this type of assistance?

References

[1] Ezoji, A., Boujut, J. F., and Pinquié, R., 2021, "Requirements for Design Reuse in Open-Source Hardware: A State of the Art," Proc. CIRP, **100**, pp. 792–797.
[2] Song, B., Zhou, R., and Ahmed, F., 2024, "Multi-modal Machine Learning in Engineering Design: A Review and Future Directions," ASME J. Comput. Inf. Sci. Eng., **24**(1), p. 010801.
[3] Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., and Tehrani, S. Y., 2025, "Research Directions for Using LLM in Software Requirement Engineering: A Systematic Review," Front. Comput. Sci., **7**, p. 1519437.
[4] Ocleppo, V., 2025, "Enhancing Requirements Engineering With Large Language Models: From Elicitation and Classification to Traceability, Ambiguity Management and API Recommendation," PhD thesis, Politecnico Di Torino, Torino, Italy.
[5] Vasudevan, P., and Reddivari, S., 2025, "The Role of Generative AI Models in Requirements Engineering: A Systematic Literature Review," Proceedings of the 2025 ACM Southeast Conference, Cape Girardeau, MO, Apr. 24–26, ACM, pp. 188–194.
[6] Saloranta, S., 2024, "From Documents to Models: Adopting MBSE Practices in Complex Embedded Systems and Leveraging Large Language Models," MSc thesis, University of Turku, Turku, Finland.
[7] Assres, G., Bhandari, G., Shalaginov, A., Gronli, T.-M., and Ghinea, G., 2025, "State-of-the-Art and Challenges of Engineering ML-Enabled Software Systems in the Deep Learning Era," ACM Comput. Surv., **57**(10), pp. 1–35.
[8] Khan, J. A., Qayyum, S., and Dar, H. S., 2025, "Large Language Model for Requirements Engineering: A Systematic Literature Review."
[9] Rosa, S., 2025, "Large Language Models for Requirements Engineering," PhD thesis, Politecnico di Torino.
[10] Sapkota, R., Cao, Y., Roumeliotis, K. I., and Karkee, M., 2025, "Vision-Language-Action Models: Concepts, Progress, Applications and Challenges," eprint arXiv:2505.04769[cs].
[11] Sood, D., and Sprenger, M., 2024, "AutoComply: Automating Requirement Compliance in Automotive Integration Testing," MSc thesis, University of Gotenburg, Gotenburg, Sweden.
[12] Zhang, W., Joseph, J., Chen, Q., Koz, C., Xie, L., Regmi, A., Yamakawa, S., Furuhata, T., Shimada, K., and Kara, L. B., 2024, "A Data Augmentation Method for Data-Driven Component Segmentation of Engineering Drawings," ASME J. Comput. Inf. Sci. Eng., **24**(1), p. 011001.
[13] Shah, J. J., Ramnath, S., Menzel, S., Rios, T., Kocer, F., Whalen, E., Pajot, J., Adrian, A., and Kumar, P., 2025, "Principles and Metrics for Curating Large Engineering Simulation Data Sets for ML," ASME J. Comput. Inf. Sci. Eng., **25**(12), p. 120811.
[14] Fang, W., Wang, J., Lu, Y., Liu, S., Wu, Y., Ma, Y., and Xie, Z., 2025, "A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA," eprint arXiv:2504.03711[cs].
[15] Pan, J., Zhou, G., Chang, C.-C., Jacobson, I., Hu, J., and Chen, Y., 2025, "A Survey of Research in Large Language Models for Electronic Design Automation," ACM Trans. Des. Autom. Electron. Syst., **30**(3), pp. 1–21.
[16] Lin, R., Ramesh, R., Iannopollo, A., Sangiovanni Vincentelli, A., Dutta, P., Alon, E., and Hartmann, B., 2019, "Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, May 4–9, ACM, pp. 1–13.
[17] Luukko, M., 2024, "Utilization of Artificial Intelligence in Electronics Design," MSc thesis, Tampere University, Tampere, Finland.
[18] Chen, L., Chen, Y., Chu, Z., Fang, W., Ho, T.-Y., Huang, R., Huang, Y., et al., 2024, "Large Circuit Models: Opportunities and Challenges," Sci. China Inform. Sci., **67**(10), p. 200402.
[19] Wang, B., Zou, Z., Wan, H., Li, Y., Deng, Y., and Li, X., 2024, "An Empirical Study on the State-of-the-Art Methods for Requirement-to-Code Traceability Link Recovery," J. King Saud Univ. Comput. Inform. Sci., **36**(6), p. 102118.
[20] Calzada, P. E., Ibnat, Z., Rahman, T., Kandula, K., Lu, D., Saha, S. K., Farahmandi, F., and Tehranipoor, M., 2025, "VerilogDB: The Largest, Highest-Quality Dataset With a Preprocessing Framework for LLM-Based RTL Generation," eprint arXiv:2507.13369[cs].
[21] Zhang, Y., Yu, Z., Fu, Y., Wan, C., and Lin, Y. C., 2024, "MG-Verilog: Multi-grained Dataset Towards Enhanced LLM-Assisted Verilog Generation," 2024 IEEE LLM Aided Design Workshop (LAD), Almaden, CA, June 28–29, pp. 1–5.
[22] Ng, C. C., Lin, C. -T., Tan, Z. Q., Wang, X., Kew, J. L., Chan, C. S., and Zach, C., 2024, "When IC Meets Text: Towards a Rich Annotated Integrated Circuit Text Dataset," Pattern Recognit., **147**(C), p. 110124.
[23] Bhandari, J., Bhat, V., He, Y., Garg, S., Rahmani, H., and Karri, R., 2024, "Auto-spice: Leveraging LLMs for Dataset Creation Via Automated Spice Netlist Extraction From Analog Circuit Diagrams," arXiv-2411.
[24] Jansen, P., 2023, "From Words to Wires: Generating Functioning Electronic Devices From Natural Language Descriptions," *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, eds., Association for Computational Linguistics, pp. 12972–12990.
[25] He, Z., and Yu, B., 2024, "Large Language Models for EDA: Future Or Mirage?" Proceedings of the 2024 International Symposium on Physical Design, ACM, pp. 65–66.
[26] Chang, K., Wang, K., Yang, N., Wang, Y., Jin, D., Zhu, W., Chen, Z., et al., 2024, "Data Is All You Need: Finetuning LLMs for Chip Design Via an Automated Design-Data Augmentation Framework," Proceedings of the 61st ACM/IEEE Design Automation Conference, San Francisco, CA, June 23–27, pp. 1–6.
[27] Uyar, E. B., Gürsoy, A. E., Gökce, C., and Temizel, T. T., 2025, "Low-Level Hardware Requirement Classification Using Large Language Models: Challenges, Insights, and Future Directions for Embedded Control Systems," Joint Proceedings of REFSQ-2025 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located With REFSQ 2025.
[28] Amiraz, C., Cuconasu, F., Filice, S., and Karnin, Z., 2025, "The Distracting Effect: Understanding Irrelevant Passages in RAG," Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, July 27–Aug. 1, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds., Association for Computational Linguistics, pp. 18228–18258.
[29] Wang, H., Prasad, A., Stengel-Eskin, E., and Bansal, M., 2025, "Retrieval-Augmented Generation With Conflicting Evidence," Proceedings of Conference of Language Modeling (COLM), eprint arXiv:2504.13079[cs].
[30] Barbiero, L., Agostini, F., Baci, E., Frigo, F., Vianello, M., Pozza, D., and Campese, S., 2025, "Domain-Specific and Cross-Lingual Synthetic Data Generation for Information Retrieval Training in RAG Applications," *Intelligent Systems and Applications*, K. Arai, ed., Vol. 1567, Lecture Notes in Networks and Systems, Springer Nature Switzerland, Cham, pp. 514–536.
[31] Elburg, J. v., Putten, P. v. d., and Marx, M., 2025, Can We Evaluate RAGs With Synthetic Data?, e-print arXiv:2508.11758[cs].
[32] Skelic, L., Xu, Y., Cox, M., Lu, W., Yu, T., and Han, R., 2025, "CIRCUIT: A Benchmark for Circuit Interpretation and Reasoning Capabilities of LLMs," e-print arXiv:2502.07980[cs].
[33] Lu, Z., Xu, H., Chen, A., Tang, S., Zhang, J., Feng, Y., Pan, W., and Huang, J., 2025, "Hsg-Rag: Hierarchical Knowledge Base Construction for Embedded System Development," ACM Trans. Des. Autom. Electron. Syst., **30**(6), pp. 1–26.
[34] Kashyap, P., Rouf, N., Choi, Y., Franzon, P., and Cheng, C., 2025, "Rag-Em: Retrieval-Augmented Generation for Electromagnetic System Design," 2025 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC + SIPI), Raleigh, NC, Aug. 18–22, IEEE, pp. 158–163.
[35] Bai, K., Yan, P., Liu, L., and Jia, T., 2025, "Verirag: Design AI-Specific CPU Co-processor With Rag-Enhanced LLMs," 2025 International Symposium of Electronics Design Automation (ISEDA), Hong Kong, May 9–12, pp. 76–82.
[36] Xiao, Z., He, X., Wu, H., Yu, B., and Guo, Y., 2025, "Eda-Copilot: A Rag-Powered Intelligent Assistant for Eda Tools," ACM Trans. Des. Autom. Electron. Syst., **30**(6), pp. 1–24.
[37] Wang, Y., Dai, G., Ke, S., and Zheng, C., 2024, "Evaluating Sparse and Dense Retrieval in Retrieval-Augmented Generation Systems: A Study," Proceedings of the 2024 10th International Conference on Communication and Information Processing, Hainan, China, Nove. 21–24, ACM, pp. 548–554.
[38] Li, X., Lipp, J., Shakir, A., Huang, R., and Li, J., 2024, "BMX: Entropy-Weighted Similarity and Semantic-Enhanced Lexical Search," eprint arXiv:2408.06643[cs].
[39] Luan, Y., Eisenstein, J., Toutanova, K., and Collins, M., 2021, "Sparse, Dense, and Attentional Representations for Text Retrieval," Trans. Assoc. Comput. Linguist., **9**, pp. 329–345.
[40] Zhuang, S., Ma, X., Koopman, B., Lin, J., and Zuccon, G., 2024, "PromptReps: Prompting Large Language Models to Generate Dense and Sparse Representations for Zero-Shot Document Retrieval," eprint arXiv:2404.18424[cs].
[41] Shi, Y., Tao, Z., Gao, Y., Zhou, T., Chang, C., Wang, Y., Chen, B., et al., 2025, "AMSnet-KG: A Netlist Dataset for LLM-Based AMS Circuit Auto-design Using Knowledge Graph RAG," ACM Trans. Des. Autom. Electron. Syst., **30**(6), p. 3736166.
[42] Formal, T., Lassance, C., Piwowarski, B., and Clinchant, S., 2022, "From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective," Proceedings of the 45th International ACM SIGIR

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026

Conference on Research and Development in Information Retrieval, SIGIR'22, Association for Computing Machinery, pp. 2353–2359.
[43] Pustejovsky, J., and Stubbs, A., 2013, "Natural Language Annotation for Machine Learning," O'Reilly Media, Sebastopol, CA OCLC: ocn794362649.
[44] Anthilla, 2024, "Anthilla/AnthC," Feb. publication-date: 2021-06-03T14:16:46Z.
[45] Fang, L., Zhang, J., Zong, H., Wang, X., Zhang, K., Shen, J., and Lu, Z., 2023, "Open-Source Lower Controller for Twelve Degrees of Freedom Hydraulic Quadruped Robot With Distributed Control Scheme," HardwareX, **13**, p. e00393.
[46] Klar, V., Pearce, J. M., Kärki, P., and Kuosmanen, P., 2019, "Ystruder: Open Source Multifunction Extruder With Sensing and Monitoring Capabilities," HardwareX, **6**, p. e00080.
[47] Lau, S. K., Ribeiro, F. A., Subbiah, J., and Calkins, C. R., 2019, "Agenator: An Open Source Computer-Controlled Dry Aging System for Beef," HardwareX, **6**, p. e00086.
[48] Bujnowicz, L., and Sarewicz, M., 2022, "Multichannel Pulse High-Current Driver of Magnetic Actuator," HardwareX, **11**, p. e00286.
[49] Sabah Abbas, N., Salim, M. S., and Sabri, N., 2024, "ASCD: Automatic Sensing and Control Device for Crop Irrigation Scheduling," HardwareX, **18**, p. e00523.
[50] Poulsen, E., Eggertsen, M., Jepsen, E. H., Melvad, C., and Rysgaard, S., 2022, "Lightweight Drone-Deployed Autonomous Ocean Profiler for Repeated Measurements in Hazardous Areas – Example From Glacier Fronts in NE Greenland," HardwareX, **11**, p. e00313.
[51] Lezcano, H., Rodas, J., Pacher, J., Ayala, M., and Romero, C., 2023, "Design and Validation of a Modular Control Platform for a Voltage Source Inverter," HardwareX, **13**, p. e00390.
[52] Driouich, I., Cao, H., and Thomas, E., 2025, "Diverse and Private Synthetic Datasets Generation for RAG Evaluation: A Multi-agent Framework," eprint arXiv:2508.18929[cs].
[53] Formal, T., Piwowarski, B., and Clinchant, S., 2021, "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking," Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, July 11–15, ACM, pp. 2288–2292.
[54] Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., and Liu, Z., 2024, "M3-Embedding: Multi-linguality, Multi-functionality, Multi-granularity Text Embeddings Through Self-Knowledge Distillation," Findings of the Association for Computational Linguistics: ACL 2024, L.-W. Ku, A. Martins, and V. Srikumar, eds., Association for Computational Linguistics, pp. 2318–2335.
[55] Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., et al., 2021, "Milvus: A Purpose-Built Vector Data Management System," Proceedings of the 2021 International Conference on Management of Data, Shanghai, China, July 4–5, ACM, pp. 2614–2627.
[56] Li, C., Liu, Z., Xiao, S., and Shao, Y., 2023, "Making Large Language Models a Better Foundation for Dense Retrieval," eprint arXiv:2312.15503[cs].
[57] Rice, R. E., Aagarwal, S. P., and Kortum, P. T., 2023, "Effects of Task Difficulty and Presentation Order in Subjective Usability Measurement," Proc. Human Factors Ergon. Soc. Ann. Meet., **67**(1), pp. 2168–2172.
[58] Hart, S. G., and Staveland, L. E., 1988, "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," *Advances in Psychology*, Vol. 52, North-Holland, pp. 139–183.
[59] Moreira, G. d. S. P., Ak, R., Schifferer, B., Xu, M., Osmulski, R., and Oldridge, E., 2024, "Enhancing Q&A Text Retrieval With Ranking Models: Benchmarking, Fine-tuning and Deploying Rerankers for RAG," Proceedings of the CIKM Workshop: GenAI and RAG Systems for Enterprise, eprint arXiv:2409.07691[cs].
[60] Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., et al., 2024, "Searching for Best Practices in Retrieval-Augmented Generation," Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Miami, FL, Nov. 12–16, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, eds., Association for Computational Linguistics, pp. 17716–17736.
[61] Abdallah, A., Mozafari, J., Piryani, B., Ali, M., and Jatowt, A., 2025, "From Retrieval to Generation: Comparing Different Approaches," eprint arXiv:2502.20245[cs].
[62] Liang, K.-Y., and Zeger, S. L., 1986, "Longitudinal Data Analysis Using Generalized Linear Models," Biometrika, **73**(1), pp. 13–22.
[63] Lin, S., Hua, W., Li, L., Wang, Z., and Zhang, Y., 2025, "ADO: Automatic Data Optimization for Inputs in LLM Prompts," Findings of the Association for Computational Linguistics: ACL 2025, Vienna, Austria, July 27–Aug. 1, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds., Association for Computational Linguistics, pp. 26134–26146.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/26/7/071008/7591582/jcise-25-1559.pdf by Orta Dogu Teknik Universitesi-METU user on 04 August 2026