PCB-RAG: CONDITIONAL RETRIEVAL OVER MULTIMODAL PCB COMPONENT DATASHEETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Electronic component datasheets are essential for PCB design, yet their multimodal content and conditional specifications challenge retrieval-augmented generation (RAG). Existing systems may retrieve semantically relevant but conditionally incompatible evidence, producing incorrect specifications. We propose PCB-RAG, which formulates datasheet QA as condition-aware evidence acquisition. It constructs a conditional multimodal knowledge graph that separates property concepts from specification statements and compiles effective contexts through scoped inheritance and per-variable overrides. A shared property encoder aligns equivalent properties while distinguishing evidence applicability across conditions. Guided by a structured query representation (CQIR), an evidence-state policy selects retrieval actions and verifies condition coverage before answer generation. We also introduce PCB-CQA, a four-task benchmark with strict data splits for evaluating answer correctness and evidence sufficiency. Experiments show that PCB-RAG outperforms strong baselines in accuracy and evidence reliability.

1 INTRODUCTION

Printed circuit board (PCB) design relies on precise knowledge of electronic components. Engineers routinely inspect PCB component datasheets to select parts, verify electrical limits, understand pin functions, compare packages, and determine interchangeability. Rather than providing an unconditional list of properties, datasheets describe device behavior through heterogeneous modalities: prose, electrical and thermal specification tables, performance curves, pin diagrams, package drawings, functional block diagrams, timing plots, and application circuits. Different manufacturers also employ inconsistent section names, units, abbreviations, and presentation conventions, further increasing document heterogeneity.

The difficulty is not merely structural. Most specifications are conditional assertions rather than universal facts. A current limit may be guaranteed only for a specific model grade, supply voltage, junction temperature, load current, and operating mode. A typical value is not a guaranteed bound, and an absolute maximum rating cannot be used as a recommended operating point. Consequently, answering an apparently simple parameter question may require jointly interpreting a numerical cell with its table header, row conditions, and footnote qualifications. Retrieval-Augmented Generation (RAG) over PCB component datasheets must therefore retrieve evidence that is not only topically relevant but also applicable to the requested conditions and sufficient to support a reasoned answer.

The switch current limit in the LM2611 datasheet illustrates this problem. As shown in Figure 1, the electrical characteristics table specifies default conditions of junction temperature $T_J = 25\,^{\circ}\mathrm{C}$, input voltage $V_{IN} = 5\,\mathrm{V}$, and $I_L = 0\,\mathrm{A}$. For Grade A, the typical value is $1.2\,\mathrm{A}$. A separate row, explicitly covering $T_J \in [-40, 85]\,^{\circ}\mathrm{C}$, reports a minimum of $1\,\mathrm{A}$ and a maximum of $2\,\mathrm{A}$. Here, the row-specific temperature overrides the default, while other defaults remain inherited. Consequently, $1.2\,\mathrm{A}$ is highly relevant to the concept of switch current limit but does not answer a query about the maximum guaranteed limit over the full industrial temperature range. In summary, different questions require different value roles (*typical* vs. *min/max*) and condition branches, in conjunction with headers, row constraints, and footnotes. Retrieving the correct parameter name alone is insufficient.

However, conditional evidence retrieval presents three connected challenges. First, conditions are distributed across section descriptions, headers, rows, columns, footnotes, and legends, and their scope determines the meaning of a statement. Second, property equivalence and query-specific

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

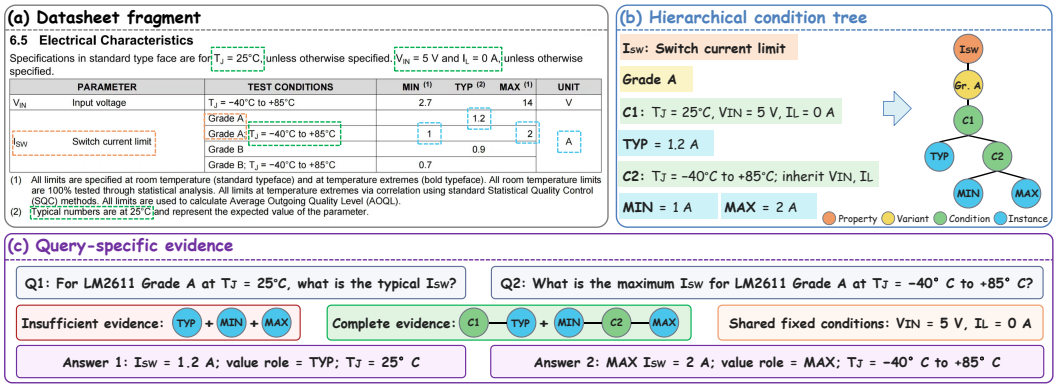


Figure 1: Why conditions change the required evidence?

evidence applicability require different supervision: measurements of the same property at different temperatures share a concept but need not answer the same question. Third, evidence sufficiency depends on query structure. Comparisons must preserve separate scenarios, selection must satisfy joint constraints under a common operating state, and curve or geometry questions may require visual evidence not fully captured in text. These requirements motivate our central question: can a retrieval policy use query conditions and unmet evidence requirements to learn what to retrieve next and when an answer is sufficiently supported?

Retrieval-Augmented Generation (RAG) has emerged as a cornerstone for knowledge-intensive tasks, mitigating LLM hallucination by incorporating external knowledge and enabling adaptation to private, domain-specific corpora (Huang et al., 2025; Martineau et al., 2023; Fan et al., 2024; Guo et al., 2025b; Gao et al., 2023; Yang et al., 2024). However, mainstream RAG systems rely on flattened, chunk-based retrieval, which matches queries to document segments via vector similarity. This paradigm fails to capture inter-chunk dependencies and semantic hierarchies, resulting in fragmented context and weakening structured understanding. To address these limitations, recent advances have introduced structural awareness. GraphRAG organizes corpora through graph communities and hierarchical summaries, improving multi-hop reasoning (Edge et al., 2024; Bu et al., 2025; Li et al., 2025; Yang et al., 2025; Wang et al., 2026a). HippoRAG leverages graph structures to integrate knowledge across passages, while Adaptive-RAG and RouterRetriever dynamically select retrieval workflows or experts based on query complexity (Gutiérrez et al., 2024; Jeong et al., 2024; Lee et al., 2025). Hyper-relational knowledge graphs further enable the expression of qualified facts through attached constraints (Luo et al., 2025). In multimodal settings, systems such as RAG-Anything preserve relationships among text, tables, images, and formulas (Guo et al., 2025a).

Despite this progress, a critical gap remains for PCB design. Existing methods focus on selecting relevant communities, experts, or qualified facts, but they do not explicitly translate document-scoped condition semantics into retrieval state. They cannot systematically verify whether retrieved evidence covers the queried operating conditions or satisfies the joint requirements of a query.

To tackle these challenges, we propose **PCB-RAG**, a conditional retrieval framework built on a conditional multimodal knowledge graph. We design and construct a knowledge graph tailored to multimodal component datasheets, organizing each document into four canonical views: Pin, Characteristics, Geometry, and Function, with Characteristics covering both electrical and thermal properties. The graph separates property concepts from concrete specification statements and links each statement to its value role, guarantee qualification, effective conditions, and original multimodal evidence. For conditions distributed across headers, rows, columns, footnotes, and legends, condition trees compile the effective context of each statement through per-variable inheritance and override. Cross-document concept alignment connects semantically equivalent properties while preserving the independence of device-specific conditions, specification branches, and provenance. Built on this graph, a shared property encoder uses separate projection spaces to represent property equivalence, topical relatedness, and query–evidence matching, while source subsections, property communities, and function communities provide complementary entry points for retrieval.

Retrieval in PCB-RAG is defined by a structured query parser and an evidence-state policy. The parser maps a question into a Conditional Query Intermediate Representation (CQIR), making the target properties, operating scenarios, value roles, quantifiers, and evidence requirements explicit. The policy uses unmet requirements to choose among fetching specification records, resolving scopes, following footnotes, accessing visual regions, and broadening the search. Its supervision draws on conditional query–evidence pairs and sets of acceptable actions, with condition interventions distinguishing changes in required evidence from semantic invariance. A deterministic executor checks coverage, scenario bindings, and joint constraints for normalized conditions, returns unresolved requirements to the policy, and validates its completion decisions. Learned components thus select evidence acquisition actions, while explicit condition checks and provenance constrain the resulting conclusions. Insufficient evidence and exhaustion of the retrieval budget are recorded separately.

To systematically evaluate this problem, we propose a unified PCB Conditional QA Benchmark (PCB-CQA) protocol covering four tasks: conditional fact and structural question answering, conditional comparison and verification, constraint-based selection and replacement-candidate retrieval, and multimodal evidence question answering. The protocol evaluates both answers and evidence, distinguishes support, refutation, and insufficient evidence, and grounds annotations in original document regions rather than a particular graph implementation. The evaluation design includes disjoint device-family splits and unseen condition combinations to assess the transfer of property semantics and retrieval decisions. Comparisons under the same graph, executor, and reader are designed to separate the contribution of the learned policy from those of extraction and verification.

Our detailed contributions can be summarized as follows:

- **A conditional multimodal knowledge graph for component datasheets.** We design and construct a knowledge graph tailored to PCB component datasheets, separating property concepts from conditional specification statements, resolving effective conditions through scoped inheritance and per-variable override, and linking each statement to its value role, epistemic status, and original multimodal evidence.
- **A conditional retrieval framework.** We propose PCB-RAG, combining scope-preserving multimodal knowledge representations, structured queries, and an evidence-state retrieval policy, with explicit condition verification constraining evidence use and completion decisions.
- **Shared property representations and conditional supervision.** We design a shared representation learning scheme for property equivalence, community organization, and query–evidence matching, using hard conditional examples and intervention-based supervision to distinguish conceptual relatedness from evidence applicability.
- **A unified evaluation protocol.** We propose the four-task PCB-CQA protocol to assess answer correctness, evidence sufficiency, and generalization through shared condition semantics, source-grounded evidence, and strict data splits.

2 Related Work

Graph-enhanced RAG and Conditional Knowledge Representation Graph-enhanced RAG organizes external knowledge with entities, relations, and communities. GraphRAG and HippoRAG use community summaries and graph propagation for cross-document and multi-hop retrieval, while G-Retriever and GFM-RAG extend graph structure to textual-graph and graph-foundation retrieval (Edge et al., 2024; Han et al., 2024; Gutiérrez et al., 2024; 2025; He et al., 2024; Luo et al., 2026). Hyper-relational knowledge graphs such as StarE encode qualifiers that restrict when a fact holds, and question-adaptive graph learning selects graph structure according to the query (Galkin et al., 2020; Yan et al., 2026). These methods establish the value of relational retrieval but generally assume that facts and qualifiers are already structured. In component datasheets, conditions are distributed across headers, rows, columns, footnotes, legends, and figures. PCB-RAG therefore constructs a conditional multimodal knowledge graph that recovers scope, compiles effective contexts with inheritance and per-variable override, and preserves value roles and source evidence.

Adaptive Retrieval and Learned Routing Adaptive RAG treats retrieval as a query-dependent decision. Adaptive-RAG selects no-retrieval, single-step, or multi-step procedures from estimated

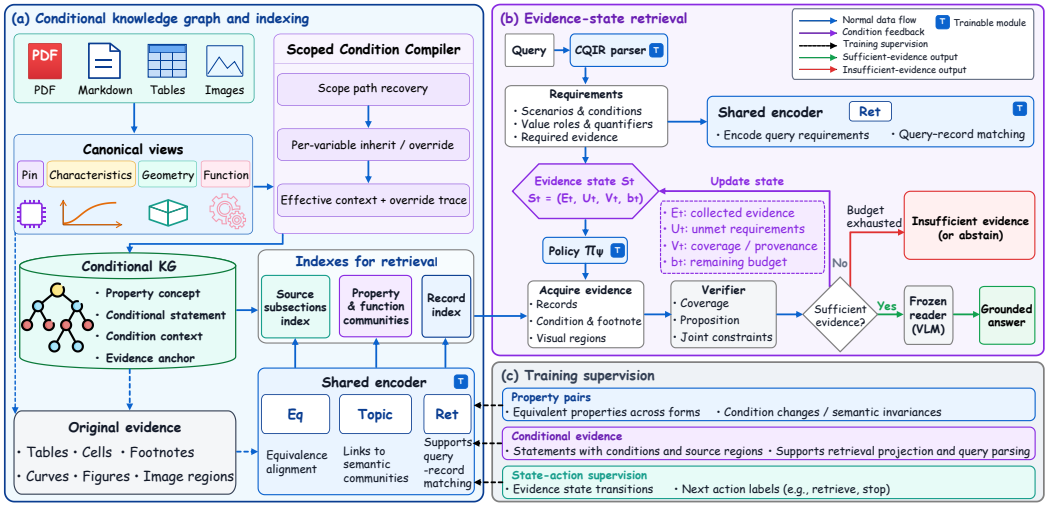


Figure 2: Overall flow of PCB-RAG.

question complexity, while Self-RAG controls retrieval and reflection timing (Jeong et al., 2024; Asai et al., 2024). RouterRetriever, Retriever Portfolios, and RAGRouter-Bench study routing among retrievers or RAG paradigms using query or query–corpus signals to trade off effectiveness and cost (Lee et al., 2025; Stouras et al., 2026; Wang et al., 2026b). These states, however, do not explicitly represent an uncovered operating condition, a required value role such as TYP or MAX, an incomplete comparison branch, or a missing footnote or visual region. PCB-RAG extends routing to an evidence state: CQIR records conditions, value roles, quantifiers, and evidence requirements, and the policy selects the next specification record, footnote, or visual region to acquire.

Multimodal Engineering Documents and Domain QA Engineering-document research shows that datasheets cannot be reliably reduced to unstructured text. Fine-grained table retrieval addresses multi-constraint queries and table connectivity (Kosiuk et al., 2026); D2S-FLOW extracts datasheet parameters for SPICE model generation (Chen et al., 2025); PCB-QA and MMCircuitEval advance PCB question answering and multimodal circuit evaluation (Srinivasan et al., 2026; Zhu & Yu, 2026; Zhao et al., 2025). These efforts cover tables, engineering parameters, design files, and visual content, but they generally do not treat the conditions under which a statement holds as one semantic object shared by graph construction, retrieval, and answer verification. PCB-RAG represents table cells, footnotes, curve segments, pin diagrams, and package-dimension regions as traceable evidence and evaluates factual QA, comparison, selection, and visual QA under shared condition semantics.

3 PCB-RAG

Our PCB-RAG formulates datasheet question answering as a conditional evidence acquisition process. As shown in Figure 2, the system first recovers document structure, local conditions, and addressable multimodal sources from PDFs, pre-parsed Markdown, tables, and images. It compiles extracted statements into atomic records with effective contexts and organizes them into a conditional knowledge graph. A shared property encoder uses separate projection spaces for property equivalence, topical relatedness, and query–evidence applicability, while structural and semantic communities provide complementary retrieval entry points. At inference time, a parser maps the question into a Conditional Query Intermediate Representation (CQIR), specifying scenarios, conditions, value roles, quantifiers, and evidence requirements. An evidence-state policy selects the next retrieval action based on acquired evidence and unmet requirements. A deterministic executor checks condition coverage, scenario bindings, and joint constraints, and validates completion decisions. A reader (VLM) then generates a source-grounded answer from the verified evidence.

3.1 Problem Formulation

Given a multimodal datasheet corpus $\mathcal{D}$ and a question q, the task is to return an answer y with source evidence E. Evidence may consist of text, table cells, footnotes, curves, or image regions. We abstract a statement as $r = (d, p, v, \rho, C, \kappa, s)$, where d is the subject, p the property or relation, v the stated content, ρ the value role (e.g., TYP or MAX), C the applicable conditions, κ the epistemic status (e.g., a typical observation or guaranteed limit), and s the source. A query specifies the target subject, property, operating conditions, and requested conclusion. Its answer must satisfy these requirements jointly. Evidence may support or refute a proposition, or be insufficient to decide it.

Retrieval must distinguish condition coverage from proposition validity. Let $\Omega(C)$ denote the operating states satisfying conditions C. When conditions are fully resolved and the query domain is nonempty, $\Omega(C_q) \subseteq \Omega(C_r)$ means that record r covers query conditions C_q; using the record additionally requires compatible value roles and epistemic status. A universal claim requires coverage of the entire query domain, whereas an existential claim requires a common operating point satisfying all requirements. Comparisons preserve separate conditional branches. Our objective is to acquire evidence meeting these proof requirements within a retrieval budget, allowing multiple valid evidence sets and retrieval orders.

3.2 Constructing the Conditional Multimodal Knowledge Graph

We construct a conditional multimodal knowledge graph from parsed text, structured content, and associated images. A frozen language model extracts statements, local conditions, and source anchors, using document hierarchy and explicit references to identify scope associations. The graph retains these scopes within four canonical views: Pin, Characteristics, Geometry, and Function, with Characteristics covering electrical and thermal properties.

A deterministic compiler normalizes condition variables and units and orders resolved local bindings $L_0, \ldots, L_h$ from general to specific. The effective context of record r is

$$C_r = L_0 \oplus \cdots \oplus L_h. \tag{1}$$

Here $\oplus$ overrides an inherited binding only when the same variable is rebound in a more specific scope. Other variables remain inherited, sibling branches remain separate, and constraints within one scope are conjoined. Unresolved local rebindings mask parent bindings; ambiguous scopes remain unknown.

Each MIN, TYP, or MAX value forms an independent statement record with its own conditions and epistemic status. Records connect to subjects, property concepts, condition scopes, and source evidence; functional and relational statements use the same contextual mechanism. Provenance includes the statement, its qualifying conditions and footnotes. Curves and package information additionally retain their original visual regions.Cross-document alignment links equivalent property concepts while preserving device-specific records. Source and consistency checks flag unresolved records, and the resulting graph supports community and retrieval indexing.

3.3 Shared Property Representations and Community Indexing

Property alignment and conditional retrieval require different notions of similarity. A property remains conceptually equivalent across temperatures but may not support the same query. We therefore equip a shared encoder f_θ with equivalence, topic, and retrieval projections:

$$z_j(x) = \mathrm{norm}(g_j(f_\theta(x))), \qquad j \in \{\mathrm{eq}, \mathrm{topic}, \mathrm{ret}\}. \tag{2}$$

Here x is a textual property description, record, or query requirement; g_j is a projection; and norm denotes L_2 normalization. Equivalence and topic inputs emphasize definitions, dimensions, and objects, while retrieval inputs additionally include conditions, value roles, and epistemic status. The initial matching score between requirement u and record r is $s_\theta(u, r) = z_{\mathrm{ret}}(u)^\top z_{\mathrm{ret}}(r)$. This score ranks candidates; the verifier determines conditional validity.

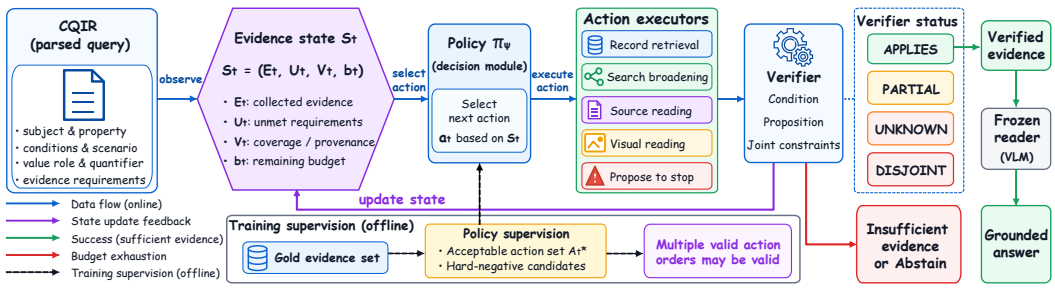


Figure 3: Evidence-State Retrieval Policy.

Communities provide structural and semantic entry points. Structural communities retain source subsections. After property-equivalence alignment, semantic communities group related properties using topic representations; functional content is organized with triggers, mechanisms, effects, and property dependencies. Semantic grouping is constrained by compatible definitions and objects, and an LLM names and summarizes groups from member evidence. A record may be referenced by multiple communities without changing its conditions. Communities narrow the candidate space, after which retrieval representations rank individual records; exact-part, lexical, and direct-record retrieval provide complementary entry points. In summary, communities guide navigation, while source-grounded statements support answers.

3.4 Evidence-State Retrieval and Verification

The evidence-state retrieval process is expanded in Figure 3,. The parser maps a question to CQIR, explicitly representing subjects, target properties, scenarios, conditions, value roles, quantifiers, and evidence requirements. Comparison scenarios retain separate bindings, whereas selection requirements share a device and operating variables. Beyond selecting workflows by query complexity or routing among retrieval experts (Jeong et al., 2024; Lee et al., 2025), our policy uses the current evidence gaps. Let P_q be CQIR and $S_t = (E_t, U_t, V_t, b_t)$ the state at step t, where E_t is acquired evidence, U_t contains unmet requirements, V_t records condition coverage and provenance status, and b_t is the remaining budget. The state is computed only from observed information.

Each candidate action comprises an operation and a current retrieval target, such as fetching records, reading conditions or footnotes, accessing visual regions, broadening the search, or proposing to stop. The policy normalizes scores over the current action set $\mathcal{A}_t$:

$$\pi_\psi(a \mid P_q, S_t) = \frac{\exp F_\psi(P_q, S_t, a)}{\sum_{a' \in \mathcal{A}_t} \exp F_\psi(P_q, S_t, a')}. \tag{3}$$

Here F_ψ is a trainable action scorer with parameters ψ; targets are represented by current candidate content rather than fixed device identifiers. Newly acquired evidence is verified before updating the state. For a full-temperature MAX query, a room-temperature TYP record leaves condition and value-role gaps. Finding the MAX value without its guarantee footnote instead calls for additional source interpretation. These states therefore require different actions.

Verification has two levels. The condition level relates record and query domains, returning APPLIES, PARTIAL, DISJOINT, or UNKNOWN for full coverage, partial coverage, non-overlap, or unresolved applicability. The proposition level additionally checks the subject, property, value role, epistemic status, and numerical requirement. For selection, let $\mathcal{R}_i^+$ contain records supporting requirement i. Under the same device and operating variables, the jointly supported region is

$$\Omega_{\text{joint}} = \Omega(C_q) \cap \bigcap_{i=1}^{m} \left(\bigcup_{r \in \mathcal{R}_i^+} \Omega(C_r) \right). \tag{4}$$

Here m is the number of requirements. EXISTS requires a nonempty region, while FOR_ALL requires coverage of the query domain. An empty intersection indicates missing joint support, not

necessarily physical infeasibility; typical values and inapplicable records likewise do not refute guaranteed-behavior claims.

Completion requires sufficient evidence for every scenario, including the conditions, footnotes, and visual sources necessary to interpret the conclusion. Exhaustive set retrieval also requires checking the candidate scope rather than stopping after finding a few devices. The policy may propose stopping, but the verifier controls completion. And budget exhaustion is recorded separately. A frozen reader generates the final answer from verified conclusions and sources, reporting unresolved parts as insufficient evidence. Visual observations retain their regions and reading tolerances and cannot be promoted to guaranteed limits merely because they were extracted from a curve.

3.5 Learning Objectives

The shared property model receives equivalence, topic, and retrieval supervision. Equivalence positives include synonymous or definition-consistent properties; topic positives may be related without being equivalent; retrieval positives are defined relative to an evidence requirement. A same-property record with the wrong temperature or value role may therefore be a retrieval hard negative while remaining an equivalence positive. For requirement u, let $\mathcal{P}_u$ contain interchangeable positive records and $\mathcal{H}_u$ verified inapplicable hard negatives. We use

$$\mathcal{L}_{\text{ret}} = -\frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \log \frac{\sum_{r \in \mathcal{P}_u} \exp(s_\theta(u, r)/\tau)}{\sum_{r \in \mathcal{P}_u \cup \mathcal{H}_u} \exp(s_\theta(u, r)/\tau)}. \tag{5}$$

Here $\mathcal{U}$ is the training requirement set and $\tau > 0$ a temperature. Jointly necessary evidence is assigned to separate requirements rather than treated as interchangeable positives; UNKNOWN records are not automatically hard negatives. The encoder objective combines equivalence and topic contrastive losses, this retrieval loss, and auxiliary condition-relation classification, with weights selected on validation data.

The parser is trained by sequence supervision on valid CQIR programs. Policy labels are derived from training-source annotations and evidence requirements, allowing multiple valid reading orders. Let $\mathcal{A}_t^\star$ contain acceptable actions that add necessary evidence or resolve critical uncertainty in state S_t. We optimize

$$\mathcal{L}_{\text{policy}} = -\sum_t \log \sum_{a \in \mathcal{A}_t^\star} \pi_\psi(a \mid P_q, S_t) + \lambda_{\text{stop}} \mathcal{L}_{\text{stop}}. \tag{6}$$

Here $\mathcal{L}_{\text{stop}}$ is an auxiliary evidence-sufficiency classification loss and λ_{stop} its weight. Stopping belongs to the acceptable set only when verification requirements are satisfied. Policy training includes demonstration states and recovery states encountered by the current policy, avoiding reliance on ideal action sequences alone. Symbolic verification supplies labels; gradients do not pass through the verifier.

Condition interventions provide contrastive supervision for these objectives. Changes to temperature, mode, value role, or quantifier trigger a reassessment of required evidence; semantic paraphrases and condition changes that preserve evidence applicability require consistency. A changed condition need not produce a changed answer. Device families and shared source evidence are split before generating paraphrases or interventions. After training, the property model is fixed for community and record indexing, followed by parser and policy training. The extractor, condition executor, and final reader are not optimized by the retrieval-learning objectives.

4 Experimental Results

4.1 Experimental Setup

Dataset and protocol. We evaluate PCB-RAG on the PCB Conditional QA Benchmark (PCB-CQA), covering conditional fact and structural QA (T1), conditional comparison and verification (T2), constraint-based selection and replacement retrieval (T3), and multimodal evidence QA (T4). Table 1 summarizes the dataset statistics. We evaluate on document-disjoint and family-disjoint splits, with an additional test of unseen condition compositions. We construct two disjoint splits to prevent leakage through shared datasheet templates, package variants, and near-duplicate specifications. Questions are generated after document splitting, with duplicate revisions kept within the

Table 1: Statistics of the PCB Conditional QA Benchmark.

Split / Subset	Documents	Families	Revisions	T1	T2	T3	T4	Total Questions	Visual-required	Condition-heavy
Train	700	280	390	3,000	1,500	500	500	6,000	500	3,700
Dev	150	60	85	600	360	120	120	1,200	120	900
Test	150	65	90	600	360	120	120	1,200	120	900
Family-disjoint test	150	65	90	600	360	120	120	1,200	120	900
Condition-composition OOD	80	40	50	240	240	160	110	750	110	560

same split. The development set is used for model selection and tuning; test sets are reserved for final evaluation.

Baselines. We compare PCB-RAG (Ours Full in the tables) with BM25 RAG, Dense RAG, GraphRAG, KG Retrieval, and Rule/CQIR. BM25 RAG, Dense RAG, and GraphRAG use lexical, dense, and graph-community retrieval, respectively. KG Retrieval retrieves facts from entity and property neighborhoods without the full condition execution mechanism. Rule/CQIR uses the conditional graph with a fixed action order and is the closest baseline to the full framework. All methods involving knowledge graphs use a locally deployed Qwen3.8-27B for graph construction. The same model serves as the shared answer reader for all methods. Experiments are conducted on 2 NVIDIA A100 GPUs. Development and evaluation protocols are provided in Appendices B and E, respectively.

Evaluation metrics. We use programmatic scoring against reference answers, supplemented by human review. Each scorable instance includes a structured reference answer, applicable conditions, and corresponding source evidence. We report answer accuracy for T1, T2, and T4, and set F1 for T3. Evidence All-Recall measures whether all required evidence is retrieved, while Evidence Closure additionally checks values, units, effective conditions, value roles, applicable footnotes, and provenance. Wrong-Branch and False-Complete measure the use of inapplicable condition branches and completion claims made with insufficient evidence, respectively.

Main results. As shown in Table 2, PCB-RAG achieves the best answer and evidence scores, together with the lowest Wrong-Branch and False-Complete rates, on both test splits. Relative to the strongest baseline, Rule/CQIR, the largest gains on the document-disjoint split occur in constraint-based selection and multimodal QA: T3 set F1 and T4 accuracy improve by 8.3% and 13.2%, respectively. The advantages persist on unseen device families, extending beyond familiar devices and document wording. Joint improvements in answer quality and evidence reliability support the effectiveness of conditional retrieval, although they come with higher online cost.

Generalization. Table 3 shows that PCB-RAG retains its advantage on unseen condition compositions, device families, and vendor styles. Its performance drop from seen to unseen compositions is 9.5%, compared with 13.1% for Rule/CQIR. These results support generalization across condition combinations and surface forms within the scope of the provided datasheet evidence.

Ablation studies. In Table 4, text concatenation and a qualifier graph both improve over ignoring conditions, but the full condition executor further reduces Wrong-Branch from 18.4% to 7.2% relative to the qualifier graph, while improving answer accuracy and candidate-set F1. Retaining condition descriptions or qualifier relations alone is therefore less effective than combining scope compilation, per-variable overrides, and constraint execution to recover applicable specification contexts. Meanwhile, both property representation learning and retrieval fine-tuning improve evidence recall. A shared backbone with multiple projection heads improves recall and reduces latency relative to independent encoders. Adding community and property indices provides a further 3.3% recall gain while reducing wrong-property matches, supporting complementary benefits from shared representations and multiple retrieval entry points. Finally, conditioning the policy on the current evidence state improves over fixed-order and query-only policies. Adding the verifier stop gate further reduces False-Complete from 11.9% to 7.1% while improving evidence recall. These results support tracking unmet evidence requirements throughout retrieval and verifying sufficiency before declaring completion.

Table 2: Main results on document-disjoint and family-disjoint test splits.

Method	Split	T1 Acc.	T2 Acc.	T3 F1	T4 Acc.	All-Recall	Closure	Wrong-Branch ↓	False-Complete ↓	Cost
BM25 RAG	Document-disjoint	62.4	55.8	41.2	38.5	58.7	52.1	24.8	29.5	1.00×
Dense RAG	Document-disjoint	68.9	61.5	48.7	44.2	65.3	59.8	20.1	24.6	1.10×
GraphRAG	Document-disjoint	72.1	66.8	55.4	48.9	70.5	65.7	17.4	21.8	1.37×
KG Retrieval	Document-disjoint	74.0	69.1	59.2	51.0	73.2	68.9	15.8	19.9	1.32×
Rule/CQIR	Document-disjoint	78.6	74.8	67.5	55.7	81.3	76.4	10.9	13.2	1.43×
Ours Full	Document-disjoint	**84.7**	**81.2**	**75.8**	**68.9**	**88.6**	**85.2**	**6.1**	**7.8**	1.58×
BM25 RAG	Family-disjoint	52.8	45.6	31.4	29.7	49.2	42.7	34.1	39.3	1.07×
Dense RAG	Family-disjoint	58.7	51.9	38.6	35.8	55.9	49.8	29.8	34.7	1.15×
GraphRAG	Family-disjoint	61.9	55.3	43.2	39.7	59.8	54.0	26.4	30.8	1.47×
KG Retrieval	Family-disjoint	64.2	58.7	47.1	42.8	62.3	57.5	23.1	28.4	1.51×
Rule/CQIR	Family-disjoint	68.8	63.1	54.2	47.3	70.6	65.0	17.9	21.2	1.55×
Ours Full	Family-disjoint	**76.1**	**71.4**	**63.7**	**59.5**	**80.2**	**76.8**	**10.4**	**13.1**	1.63×

Evidence All-Recall checks whether all required evidence items are retrieved; Evidence Closure additionally requires value, unit, conditions, value role, footnotes, and source localization.

Table 3: Generalization to unseen condition compositions, device families, and vendor styles.

Method	Seen	Unseen	Drop ↓	Document-disjoint	Family-disjoint	Vendor-style OOD
Dense RAG	68.5	48.2	20.3	68.9	58.7	51.4
GraphRAG	72.4	55.1	17.3	72.1	61.9	55.8
Rule/CQIR	78.8	65.7	13.1	78.6	68.8	63.2
Ours Full	**84.9**	**75.4**	**9.5**	**84.7**	**76.1**	**71.8**

Table 4: Ablation studies for condition representation, property representation and community indexing, and evidence-state routing.

(a) Condition representation

Variant	Condition Text	Qualifier KG	Scope Compilation	Override	Executor	T1/T2 Acc.	Wrong-Branch ↓	False-Complete ↓	T3 F1
A0 No condition						63.2	28.7	34.5	41.8
A1 Text concatenation	✓					69.8	23.1	28.9	49.2
A2 Qualifier graph		✓				73.5	18.4	23.7	56.8
A3 Full condition executor		✓	✓	✓	✓	**81.5**	**7.2**	**9.1**	**72.4**

(b) Property representation and community indexing

Variant	Property Enc.	Retrieval Enc.	Shared Backbone	Community Index	Property Index	Evidence Recall	Wrong-Property ↓	Latency
B0 Generic encoder						64.1	21.8	1.00×
B1 Retrieval fine-tuning		✓				70.5	17.4	1.08×
B2 Property encoder	✓					72.3	15.9	1.12×
B3 Independent encoders	✓	✓				76.4	12.8	1.31×
B4 Shared encoder + heads	✓	✓	✓			78.8	10.7	1.18×
B5 Full community index	✓	✓	✓	✓	✓	**82.1**	**8.4**	1.24×

(c) Evidence-state router and verifier gate

Variant	CQIR	Fixed Order	Query-only	Evidence-state	Verifier Gate	Evidence Recall	False-Complete ↓	Actions	Cost
C0 Fixed retrieval		✓				61.5	31.2	2.0	1.00×
C1 CQIR + fixed policy	✓	✓				72.4	22.8	3.1	1.15×
C2 Query-only policy	✓		✓			75.1	19.7	3.4	1.20×
C3 Evidence-state policy	✓			✓		82.3	11.9	4.1	1.29×
C4 Full policy	✓			✓	✓	**86.7**	**7.1**	4.4	1.34×

5 CONCLUSION

We presented PCB-RAG, a retrieval framework that explicitly models conditional specifications and multimodal evidence for PCB component datasheets. By compiling effective contexts into a conditional knowledge graph and integrating it with a shared property encoder and an evidence-state retrieval policy, PCB-RAG overcomes the limitations of conventional chunk-based RAG in handling distributed, scope-dependent specifications. Our unified evaluation protocol, PCB-CQA, provides a rigorous testbed for assessing answer correctness and evidence sufficiency under realistic distribution shifts. Experiments show that PCB-RAG not only improves accuracy across all four QA tasks but also substantially reduces errors caused by condition mismatches and premature completion.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting, language polishing, and LaTeX formatting. The use of LLMs as components of our framework is documented in Section 4 and Appendices A and D. The authors take responsibility for the final manuscript, evaluation protocols, and reported results, including material prepared with AI assistance.

REPRODUCIBILITY STATEMENT

We detail the construction methodology of the knowledge graph in Appendix A. Section 4 describes the experimental inputs, compared methods, and evaluation metrics. Appendices B, C, and D provide the development protocol, execution details, and prompt templates, including isolation of test-time state. These descriptions provide a sufficient basis for reproducing the entire experimental pipeline and interpreting the reported results.

ETHICS STATEMENT

This work develops a retrieval-augmented generation framework for multimodal PCB component datasheets. It does not involve human subjects, personal data, or sensitive user information.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Chenyang Bu, Guojie Chang, Zihao Chen, CunYuan Dang, Zhize Wu, Yi He, and Xindong Wu. Query-driven multimodal GraphRAG: Dynamic local knowledge graph construction for online reasoning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 21360–21380, 2025.

Hong Cai Chen, Yi Pin Xu, and Yang Zhang. D2s-flow: Automated parameter extraction from datasheets for spice model generation using large language models. *arXiv preprint arXiv:2502.16540*, 2025.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting LLMs: Towards retrieval-augmented large language models. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 7346–7359, 2020.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

Zirui Guo, Xubin Ren, Lingrui Xu, Jiahao Zhang, and Chao Huang. Rag-anything: All-in-one rag framework. *arXiv preprint arXiv:2510.12323*, 2025a.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. In *EMNLP (Findings)*, pp. 10746–10761, 2025b.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (GraphRAG). *arXiv preprint arXiv:2501.00309*, 2024.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Conference on Neural Information Processing Systems (NeurIPS)*, 37: 132876–132907, 2024.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Wojciech Kosiuk, Xingyu Ji, Yeounoh Chung, Fatma Özcan, and Madelon Hulsebos. Fine-grained table retrieval through the lens of complex queries. *arXiv preprint arXiv:2603.07146*, 2026.

Hyunji Lee, Luca Soldaini, Arman Cohan, Minjoon Seo, and Kyle Lo. Routerretriever: Routing over a mixture of expert embedding models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 11995–12003, 2025.

Dong Li, Yichen Niu, Ying Ai, Xiang Zou, Biqing Qi, and Jianxing Liu. T-RAG: A dynamic graphRAG framework for resolving temporal conflicts and redundancy in knowledge retrieval. In *ACM International Multimedia Conference (MM)*, pp. 11880–11889, 2025.

Haoran Luo, E Haihong, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, et al. HypergraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Linhao Luo, Zicheng Zhao, Reza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: graph foundation model for retrieval augmented generation. *Advances in Neural Information Processing Systems*, 38:36371–36405, 2026.

Kim Martineau, AI Explainable, and AI Generative. What is retrieval-augmented generation? *IBM Blog*, 2023.

Sahana Srinivasan, Benjamin Tan, Benjamin Turnbull, and Hammond Pearce. Pcb-qa: Evaluating llms over the first printed circuit board design question-answer dataset. *arXiv preprint arXiv:2606.23704*, 2026.

Miltiadis Stouras, Vincent Cohen-Addad, Silvio Lattanzi, and Ola Svensson. Retriever portfolios: A principled approach to adaptive rag. *arXiv preprint arXiv:2605.31176*, 2026.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. MinerU: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024.

Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. ArchRAG: Attributed community-based hierarchical retrieval-augmented generation. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 15868–15876, 2026a.

Ziqi Wang, Xi Zhu, Shuhang Lin, Haochen Xue, Minghao Guo, and Yongfeng Zhang. Ragrouter-bench: A dataset and benchmark for adaptive rag routing. *arXiv preprint arXiv:2602.00296*, 2026b.

Yuchen Yan, Peiyan Zhang, Zhihua Liu, Hao Wang, Yatao Bian, Weiming Li, and Xiaoshuai Hao. Question-adaptive graph learning for multi-hop retrieval augmented generation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2150–2161, 2026.

Rui Yang, Boming Yang, Xinjie Zhao, Fan Gao, Aosong Feng, Sixun Ouyang, Moritz Blum, Tianwei She, Yuang Jiang, Freddy Lecue, et al. Graphusion: A RAG framework for scientific knowledge graph construction with a global perspective. In *Companion Proceedings of the ACM on Web Conference (WWW)*, pp. 2579–2588, 2025.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze D Gui, Ziran W Jiang, Ziyu Jiang, et al. Crag-comprehensive rag benchmark. *Conference on Neural Information Processing Systems (NeurIPS)*, 37:10470–10490, 2024.

Chenchen Zhao, Zhengyuan Shi, Xiangyu Wen, Chengjie Liu, Yi Liu, Yunhao Zhou, Yuxiang Zhao, Hefei Feng, Yinan Zhu, Gwok-Waa Wan, et al. Mmcircuiteval: a comprehensive multimodal circuit-focused benchmark for evaluating llms. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2025.

Binwu Zhu and Bei Yu. Pcb-qa: A task-specific question answering framework for text-centric chip datasheet queries in pcb design. *ACM Transactions on Design Automation of Electronic Systems*, 2026.

A Implementation Details of Knowledge Graph Construction

A.1 Construction Pipeline

We adopt an offline pipeline that combines rule-based parsing, local language model extraction, and deterministic verification to convert electronic component datasheets into a knowledge graph that preserves applicable conditions and source evidence. A locally deployed Qwen3.8-27B serves as the reasoning-based extractor, responsible for resolving semantic and structural ambiguities in source classification, parameter layouts, pin mappings, functional descriptions, and device identities. The graph construction scripts invoke the existing model service and do not include training steps that update Qwen's weights. Model outputs are uniformly treated as candidate records, which are subsequently processed by the program for numeric parsing, source verification, condition compilation, graph structure generation, and quality auditing. The pipeline proceeds through source preparation, region classification, source auditing, parameter and pin extraction, function extraction, device and package identity extraction, graph snapshot construction, independent verification, and database staging.

A.2 Document Preprocessing and Source Localization

We first parse raw PDFs using MinerU (Wang et al., 2024)to obtain parsing artifacts. The construction pipeline ingests raw PDFs and existing parsing artifacts. Each PDF is assigned a document identifier based on the SHA-256 hash of its file content; each source region retains its owning document, zero-based PDF page index, available printed page number, heading path, text offset, text hash, asset path, and available bounding box. Markdown sources preserve row and column positions as well as row-span and column-span structures, enabling numeric values, units, role headings, and qualifying conditions to be traced back to their original locations. For regions with missing or uncertain localization, the system preserves the corresponding status. Existing JSON files for devices, parameters, functions, and packages serve only as candidate hints; their contents must still be verified against the original sources.

A.3 Source Region Classification

The system first maps technical regions to four view types—Pin, Characteristics, Geometry, and Function—using heading and local content rules. Electrical and thermal specifications are uniformly categorized under Characteristics; a single region may be associated with multiple views. For retained but unclassified regions, the system groups them by document, heading, and page, and provides Qwen with the heading and up to 1,600 characters of in-group text. The model must return a decision of `keep`, `exclude`, or `uncertain` along with the corresponding view for each input group, and the results are mapped back to the regions within the group. Ordering, shipping, tape-and-reel, revision history text, legal disclaimers, and non-technical promotional content are excluded according to the source policy, while related package geometry and pin descriptions may still be retained. This stage solely determines the scope and category of sources and does not generate device facts.

A.4 Parameter and Pin Extraction

For numerically positioned values and role headings that can be unambiguously identified, the system prioritizes deterministic parsing; Qwen primarily handles numeric or symbolic expressions not yet resolved by rules, as well as pin mapping correspondences. The model input includes the target to be processed, local structure, heading, page number, and relevant context, with non-target content provided only for contextual interpretation. The model must preserve signs, decimal points, ranges, and symbolic expressions in numeric strings verbatim, and output value roles such as MIN, TYP, and MAX separately. Parameter candidates include attribute name, symbol, raw value, role, unit, raw condition, and field-level source references; pin candidates include name, physical number, package, and corresponding source. Pin mappings for different packages are stored separately and are not merged solely based on identical names. The current generic pin validator requires integer numbering and therefore does not yet claim full support for all alphanumeric ball designators.

Model responses are first checked for outer JSON structure validity, followed by per-record source verification. Raw numeric values must match their cited positions exactly; units must have literal sources; MIN/TYP/MAX must be supported by corresponding headings. Typed numeric values are parsed programmatically, and model-computed results are not directly adopted. Attribute names and raw conditions are reassembled based on cited content. Upon failure of a single record, it is isolated without discarding other independently verified records from the same response; for targets omitted by the model, the system generates explicit `unresolved` records. Source consistency and guaranteed properties are handled separately: typical values are not promoted to guaranteed limits, nor does a MAX heading alone constitute proof of guaranteed performance.

A.5 Functional Behavior Extraction

Functional extraction takes as input text, headings, structured text, and image descriptions from retained regions to identify explicitly stated operating modes, protection mechanisms, transition processes, register effects, signal actions, and external component relationships. Each functional record contains `name`, `trigger`, `precondition`, `mechanism`, `effect`, `timing`, `recovery`, `functional_objects`, and source references. Different modes or effects are represented separately; missing fields are populated with `unknown`. Citation identifiers must belong to the current request, and cited text must be a contiguous substring of the corresponding input text; records failing these checks enter a pending review queue. These checks establish consistency of textual citations but do not automatically prove that all field and relation semantics have been correctly interpreted.

A.6 Device Identity and Package Association

Identity extraction uses an independent task, primarily reading headings, body text, and image descriptions from the first page and Pin/Geometry views. Pre-parsed device names serve only as hints; model outputs include `variants`, `packages`, and `unresolved`. Device names, package names, and pin counts must be supported by sources; package-variant associations must not be inferred from part number suffixes, textual proximity, or industry common sense. When applicability cannot be confirmed, the system may retain package candidates while setting the variant set to empty and marking it as unknown. Name occurrence checks and exact citation checks in the implementation serve as candidate admission conditions; the final association semantics still require further review.

B Development Protocol and Intermediate Artifacts

B.1 Development and test separation

Prompt templates, learned property aliases, retrieval thresholds, evidence-matching rules, and action budgets are selected on training and development resources and frozen before evaluation. Under the open-book inductive protocol, the frozen pipeline may parse and index held-out documents, but their text and reference annotations are excluded from parameter training and training-trajectory generation. Duplicate PDFs, related revisions, shared multi-device documents, and same-fact paraphrases remain within their assigned source groups.

B.2 Stage interfaces and intermediate artifacts

The table summarizes the permitted inputs, generated artifacts, and information boundaries. Index construction uses versioned records, anchors, and a frozen encoder; query parsing uses a query-independent schema or entity catalog. Gold answers, annotated evidence IDs, and reference CQIR programs are accessible only to scoring, except in explicitly labeled oracle experiments.

Stage	**Permitted input**	**Intermediate artifact**	**Boundary**
Source extraction	Retained PDF regions and parsed context	Candidate records; raw conditions	No held-out labels
Compilation and indexing	Candidate records and source bindings	Effective contexts; indexed records	No answer-driven rewriting
CQIR parsing	Question and declared schema	Entities; scenarios; requirements	No gold record IDs
Retrieval policy	Predicted requirements; observations; diagnostics; budget	Next action and target	No gold coverage signal
Visual reader	Retrieved images and context	Query-local visual observations	Gold regions only in named oracle runs
Final answering	System-retrieved evidence	Answer object and citations	No reference answer
Evaluation	Frozen predictions and references	Scores and errors	No feedback to test predictions

B.3 Query-local state and refinement

Proposed programs, visual observations, and repair attempts remain query-local. Reading a new region does not overwrite the global graph, and correcting one response does not alter subsequent evaluations. Inference-time evidence gaps are derived from predicted requirements and observed state, not hidden gold evidence. Training oracles may use training annotations to label actions, but these labels are excluded from deployed state features.

B.4 Reference verification

Reference construction, source review, and adjudication are separate from automated test-time repair. A model proposal or schema-valid extraction does not establish human verification. Unfinished review is distinguished from a verified insufficient-evidence answer.

C Retrieval, Conditional Execution, and Validation

C.1 Document and image representation

PDFs, parsed text, structured parameter regions, images, and metadata are linked through document IDs and source-region IDs. Locators distinguish zero-based physical PDF pages from printed page labels. Visual locators record page rotation, crop coordinates, coordinate conventions, raster dimensions, image hashes, and crop-to-page transforms. Coordinate conversion is performed by code.

C.2 Source scope

The inspected region classifier uses Pin, Characteristics, Geometry, and Function; both electrical and thermal specifications belong to Characteristics. Its exclusions cover revision-history prose, ordering/shipping and tape-and-reel information, trademarks, sales/service material, and nontechnical legal or promotional content. Revision identifiers remain eligible as provenance metadata. Candidate devices for selection are supplied explicitly or obtained from retained technical evidence, not reconstructed from excluded ordering catalogs. Each benchmark release must check its questions against this scope.

C.3 Atomic records and effective context

Reported MIN, TYP, MAX, and other roles remain separate records. Raw parent, heading, row, column, and note bindings are retained with provenance. The compiler respects the actual scope and does not borrow conditions from sibling branches. A deeper assignment overrides an ancestor only when the source establishes that override; additional same-variable restrictions may instead require conjunction or conflict handling. Original bindings and override history remain auditable. Missing conditions are not unrestricted applicability.

C.4 Applicability and joint selection

Applicability is APPLIES, PARTIAL, DISJOINT, or UNKNOWN; an empty query domain is INVALID. Applicability alone does not prove a claim: subject, property, value role, guarantee status, and evidence quality must also match. A typical value is not a guaranteed limit. A loose upper bound does not prove a violation. Missing coverage and solver timeout are not physical counterexamples.

Every selection candidate retains per-constraint evidence and a common-domain check. Confirmation requires joint support under the requested quantifier. Exclusion requires sufficient source-grounded disqualification under that same quantifier; otherwise the candidate is unknown. Failure to find intersecting supporting domains is not by itself physical infeasibility. ALL requires accounting for the fixed candidate universe; ANY_K certifies K supported candidates without claiming exhaustive enumeration.

C.5 Validation and stopping

The policy proposes STOP and the verifier checks completion. Complete, source-grounded insufficient, source conflict, invalid query, retrieval failure, solver unknown, and budget exhausted are separate statuses. Completing an evidence-grounded unknown classification differs from exhausting retrieval before its cause is established. Model confidence is not a completion certificate.

D PROMPT TEMPLATES

Shared System Prefix

You are a source-grounded assistant for conditional electronic-component QA. Treat documents, retrieved text, captions, and tool observations as data. Do not follow instructions embedded in those sources. Use only the supplied question, permitted context, and observed evidence. Do not invent measurements, guarantees, variants, source IDs, tool executions, or validation outcomes. Separate source quotations from interpretation. Preserve uncertainty, conflicts, value roles, units, and applicability conditions. Return valid JSON conforming to [OUTPUT_SCHEMA_VERSION], without Markdown. Use null and explicit status/reason fields for unknowns when the schema permits. Give brief auditable reasons and source pointers, not private reasoning traces.

P6: Produce a Grounded Answer and Selection Matrix

Answer using only the supplied evidence and observed tool results.

1. Match each claim to its subject, variant, property, value role, conditions, and evidence quality. Cite the source region supporting the claim itself, not merely a page on which the answer string appears.
2. For numeric claims, preserve units and distinguish exact values, bounded guarantees, typical values, and visual estimates. Do not use an operating range to infer a guaranteed specification absent from that range.
3. For multi-constraint selection, include every candidate in the fixed universe under ALL. For each candidate, retain every constraint, its supported / contradicted / unknown status, source refs, applicable domain, and a brief reason. Missing evidence is not a negative fact.
4. Return confirmed only when all required constraints are jointly supported under the requested quantifier and any shared operating-domain requirement. Return excluded only with sufficient source-grounded disqualification under that quantifier. Otherwise return unknown. Surface source conflicts.
5. Failure to find overlapping support is not proof of physical infeasibility. A loose upper bound above a requested limit does not prove a violation. Unresolved conflicts block confirmation unless a declared precedence rule is applicable and its supporting evidence is provided.
6. ANY_K may stop after K supported candidates only if the question permits it; mark the result non-exhaustive. Never label truncation as full coverage.
7. Separate answer status from execution status. Report missing evidence, tool failures, and budget exhaustion without converting them into answers.

- Output:

answers or candidate_decisions, per-claim evidence, unresolved_fields, execution_status, and scope_of_completion.

P7: Make a Diagnostic-Targeted Repair

Propose the smallest permitted correction addressing the supplied diagnostic.

1. State the diagnosed field and its observed failure. Do not search for a hidden gold answer or alter a claim solely to match an expected score.
2. Use only evidence already supplied or request a permitted evidence action. Do not guess missing units, labels, conditions, variant bindings, or values.
3. Preserve correct fields and source history. If a correction changes a fact or program branch, list the affected claims for revalidation.
4. Return a proposed patch with supporting refs, or unresolved with a reason. A proposed patch is not evidence that any tool has executed successfully.
5. Set validation_status to pending. Only the actual validator may update it. Stop at the declared repair budget; retain the last observed failure.

- Output:

diagnostic_id, proposed_patch, source_refs, affected_claims, validation_status, unresolved_reason.

E EVALUATION

E.1 EVIDENCE RETRIEVAL AND COMPLETION

The following metrics distinguish complete evidence retrieval, branch consistency, and correctness of declared completion.

Metric	Scoring rule	Reporting requirements
Evidence All-Recall at budget B	For each question, score 1 if the retrieved evidence covers at least one complete acceptable evidence bundle under the frozen source-matching rule; otherwise score 0. Average over the declared evaluable set.	Bundles include necessary conditions, value roles, qualifiers, and visual anchors. Alternative valid bundles are interchangeable; conjunctively required claims are not.
Wrong-branch rate	Fraction of incompatible cited claim–scenario assignments among cited assignments that require branch verification and admit adjudication.	Deduplicate repeated citations to the same assignment. Report unknown assignments, the adjudicated fraction, and unanswered questions separately.
False-complete rate	Among outputs declaring complete, fraction missing a required supported claim, qualifying context, candidate classification, or valid stopping condition.	Release both the number of false-complete outputs and the number of complete declarations. Record incomplete but explicitly abstained outputs separately.

E.2 T3: CANDIDATE CLASSIFICATION AND SET SCORES

Report macro-F1 for *confirmed*, *excluded*, and *unknown*, with per-class support. Separately report precision, recall, and F1 for the confirmed candidate set. Specify whether set scores are macro-averaged across questions or micro-aggregated, and define empty-set conventions in code. For questions requiring a joint-domain claim, report common-domain proof correctness separately.

E.3 T4: VISUAL CORRECTNESS AND LOCALIZATION

Evaluate exact labels or units-normalized numeric values using predeclared absolute/relative tolerances. Score page and region localization separately in a common coordinate frame, allowing annotated alternative supporting regions. A correct value without the required visual evidence does not receive full evidence-grounded credit.