From Docs to Dialogue: Talking to Design Specifications

Maximilian Tyrchan
Stuttgart Media University
Stuttgart, Germany
mt103@hdm-stuttgart.de

Prof. Dr. Tobias Jordine
Stuttgart Media University
Stuttgart, Germany
jordine103@hdm-stuttgart.de

***Abstract*—This paper presents ChatSheetPT, a domain-specific Retrieval-Augmented Generation (RAG) system developed to improve access to complex, multimodal design specifications in datasheets. Datasheets containing design specifications of semiconductors are often lengthy collections of detailed textual descriptions, fragmented into various datasheets, and multimodal by combining tabular data and complex technical diagrams and illustrations, making them difficult to navigate and understand, particularly for non-experts. To address this challenge, ChatSheetPT integrates document classification, hybrid chunking, contextual embeddings, multimodal augmentation, and re-ranked retrieval to deliver context-aware, traceable, and multimodal answers to domain questions. The system is built in a modular way, leveraging the LangChain framework and seamlessly replaceable Large Language Models via LiteLLM. It was evaluated through LLM-as-a-judge using correctness, helpfulness, groundedness, relevance, hallucination, and conciseness metrics. In terms of helpfulness, groundedness, relevance, and hallucination, the results indicate a strong performance, while highlighting it's current limitations in conciseness and correctness. Although the exploratory nature of the evaluation, it establishes a foundation for future improvements. The findings support the assumption that RAG is an adequate approach to improve reliable and efficient access to complex technical information in the semiconductor domain.**

***Index Terms*—Retrieval-Augmented-Generation, Large Language Models, Semiconductor Development Process, Datasheets, Design-Specifications.**

I. INTRODUCTION

A. Overview of the Semiconductor Development Process

Modern semiconductor development follows a multi-stage manufacturing cycle, which includes research and development, product development and production. Although terminology can vary, the product development process generally includes the following primary phases:

1. **Design.** The design phase involves translating customer requirements into a viable model, through iterative refinement of geometry and performance modeling. The phase concludes with comprehensive design documentation, ensuring consistency in fabrication, assembly, and testing.
2. **Fabrication.** During the fabrication phase, the focus lies on establishing a stable and repeatable manufacturing process that minimizes variation, ensures product quality, and balances cost while optimizing process parameters. Key elements include material inspection, detailed process flow mapping, equipment maintenance, process monitoring, and thorough device tracking.

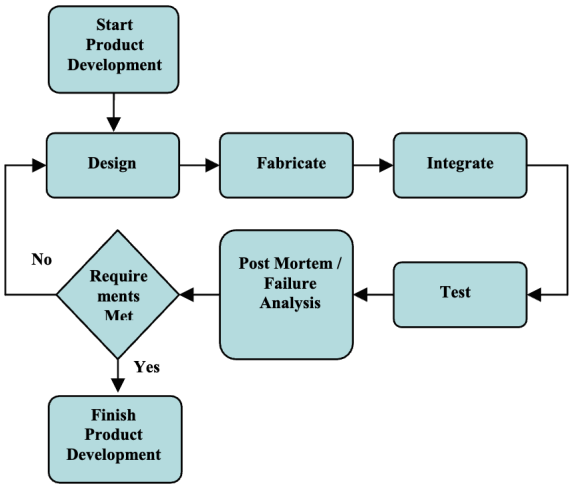

Fig. 1: Product development build cycle [1]

3. **Integration.** The integration phase in MEMS production involves assembling components through steps such as handling, bonding, interconnecting, sealing, and marking. Risks like contamination, misalignment, and poor bonding must be addressed through detailed work instructions and process controls.
4. **Testing.** The testing phase ensures the device meets customer requirements through in-process, environmental and functional, margin, and aging tests. Where the product fails to meet requirements, processes need to be refined to correct deficiencies.
5. **Post Mortem / Failure Analysis.** The post mortem phase involves disassembling and thoroughly inspection of the product for analysis. The results should be documented and reviewed for identifying root causes, prior to the next refinement cycle.

[1], [2]

B. Purpose and role of datasheets in Semiconductor Development

Design documentations created during the design phase lay the groundwork for design specifications in datasheets. Although, datasheets of design specifications vary in name and structure, they typically include the following sections: (1) Introduction, where the concepts and features of the target

product are described. (2) IO-Ports, providing detailed information about essential input and output ports. (3) Registers, description of the architecture-level registers of the design. (4) Operation, explaining the procedures for dataflow and control. (5) Architecture, which describes the high-level workflow and dataflow of the design. (6) Usage examples, offering usage examples, scenarios and illustrations. [3]
Therefore, the datasheets serve a variety of critical purposes, including:

- Providing a clear and concise description of the design intent.
- Ensuring consistent communication among design teams and stakeholders.
- Facilitating design reviews and decision-making processes.
- Serving as a reference for validation, verification, and testing procedures.
- Documenting design rationale to support future updates and modifications.

C. Problem Formulation

Despite the importance of datasheets in semiconductor development, they are often lengthy, fragmented and complex documents including tables, illustrations and mutual references that can be difficult to navigate and understand by non experts. This complexity can lead to misunderstandings, misinterpretations, and inefficiencies during and beyond the design process and hinders effective knowledge transfer. As a result, typical problems that arise include:

- During Post Mortem / Failure Analysis phase, laboratory employees regularly encounter issues with semiconductors but electronics technicians often lack the detailed knowledge of developers, requiring them to gather information from extensive documentation, in order for them to find the root cause.
- Throughout the Design phase of new ASICs, engineers often face the challenge of recalling previous solutions, but searching through hundreds of design specifications is impractical.
- New employees struggle to understand the technical aspects of products during their training due to the mentioned complexity of the design specifications.
- Customers without technical expertise have difficulty identifying the right chip for their specific requirements.

Conventional LLMs are not sufficient for this use case, as they do not dynamically incorporate external knowledge sources and therefore lack the specific information required for reliable answers [4]. This leaves a gap between the ever growing nature of datasheets and the efficient information access required by stakeholders. Therefore, this work explores how RAG can bridge that gap, enabling reliable, traceable, and context-aware interaction with technical datasheets and motivate the following research question: **RQ:** *How can a domain-specific retrieval augmented generation system be designed and implemented to improve reliable, traceable, and multimodal access to complex design specifications in semiconductor development?* The research question is based on our assumption that RAG is, to the best of our known, an adequate approach to improve reliable and efficient access to complex design specifications in datasheets.

D. Contributions

The scientific contribution of this work is to methodically design, implement, and evaluate a domain-specific RAG system to improve the accessibility to multimodal technical information in the semiconductor development process. While the developed solution does not represent a technological innovation, it rather demonstrates how the recent advances of RAG methods can be applied and deliver value to a practical, real-world industrial use-case. In summary, this work contributes the following:

- We propose a system to interact with design specifications using Retrieval-Augmented Generation (RAG) techniques by implementing a modular architecture.
- We deliver a specialized document parsing pipeline that processes multimodal datasheets into contextualized chunks for semantic retrieval.
- We present a system that allows users to ask questions about design specifications and receive accurate, traceable and multimodal responses.
- We meet the current limitations of naive RAG by applying an advanced RAG approach through contextual retrieval and reranking strategies.
- We conduct an exploratory evaluation by using LLM-as-a-judge with metrics including correctness, helpfulness, groundedness, relevance, hallucination, and conciseness to assess the functional validity.
- We identify key limitations in current RAG workflow and provide a foundation for future research in the field of applied RAG methods for the semiconductor industry.

Overall, this work demonstrates how a proof-of-concept RAG system can be tailored to domain-specific needs, showcasing the potential of RAG methods to enhance information accessibility in technical domains.

II. THEORETICAL BACKGROUND

A. Large Language Models (LLMs)

Large language models (LLMs) are large-scale, pre-trained, statistical language models based on neural networks. In recent years, rapid advances on transformer-based LLMs and further training on Web-scale text data have extended their capabilities [5]. These LLMs have demonstrated remarkable capabilities in natural language processing (NLP) and content generation [6]. Furthermore their abilities raised to instruction following, Incontext Learning and Chain of Thought. For some time these abilities were limited to text-based tasks, but recent advancements have enabled multimodal capabilities, allowing LLMs to process and generate text, images and audio [7]. Prominent examples include GPT-4o by OpenAI [8], LLaMA 3 by Meta [9], and Mistral by Mistral AI [10]. Critical problems for LLMs remain the lack of memory, stale information,

hallucinations and the fact that they are generally very large and therefore expensive to train and run [5].

B. Retrieval-Augmented Generation (RAG)

One approach to address the limitation of LLMs in accessing domain-specific information is Retrieval-Augmented Generation (RAG). RAG methods introduce a retrieval mechanism that augments the input of the LLM with context from a knowledge base, thereby enhancing the quality of the response. At its core, RAG methods consist of (a) a mechanism for retrieving contextually relevant information and (b) a guiding for the generation process of the LLM to use the retrieved information. This eliminates the need for developers to retrain the LLM on domain-specific data. Current limitations of the RAG approach are teaching the model to learn or understand new format, styles or new domains. This can be achieved by fine-tuning a LLM. However, fine-tuning is not well suited for adding new knowledge to the model. A RAG system consists of three components: Retrieval, Generation and Augmentation. [11], [12]

Retrieval. In a RAG pipeline, the retrieval component is responsible for retrieving the top-k relevant documents from a knowledge base. In order for a retriever to perform this task an accurate semantic representations of the documents and data that later will be retrieved has to be generated. Secondly, the semantic meaning of the user's query needs to alining to the semantic representations of the knowledge base. And as a third step, the retriever's output has to match the LLM's preferences in order to generate an accurate response. To accomplish this several aspects needed to be considered, such as the right chunking-strategy, a fine-tuned embedding model when applied to specific domains, a technique for handling poor user query (e.g. rewriting the query), and a method to align the retrieved output to the LLM's preferences. [12]

Augmentation. Once the chunks are retrieved from the knowledge base, the augmentation step integrates these chunks into the input (also known as context) for the LLM. This process plays a crucial role in enabling the LLM to effectively generate an adequate response.
A naive approach is to directly pass the retrieved information alongside the original query, forming an extended prompt that is passed to the LLM. However, this can lead to inefficiencies such as redundancy, incoherence, or irrelevant context. Current methods to address this issue include iterative retrieval and adaptive retrieval, allowing the model to iterate over multiple retrieval processes or adaptively adjust the retrieval strategy in certain scenarios [12]. For structured outputs, the augmentation process can also involve transforming retrieved chunks into structured formats (e.g., JSON). This allows the LLM to selectively copy relevant fields during generation, leading to more reliable and constrained outputs [13].

Generation. The generator in a RAG system is responsible for converting the retrieved information and the user's query into a coherent response. Despite the rapid advancements in LLMs, issues such as context length restrictions and their vulnerability to redundant information, persist. In order to address these challenges, research endeavors were undertaken to reprocess the gathered information during the retrieval phase. Post-retrieval processing is typically characterized by the compression of information and the subsequent re-ranking of results. [12]

Throughout the evolvement of RAG, three main paradigms have emerged:

1. Naive RAG. The naive RAG involves the traditional process of indexing, retrieval, and generation. Challenges occur in three areas: retrieval quality, response generation quality, and during the augmentation process, including low-precision, low-recall and poor integration of the retrieved context [12].
2. Advanced RAG. In order to improve the quality of the retrieval generation, Advanced RAG incorporates preretrieval and post-retrieval methods such as sliding window, fine-grained segmentation, and metadata [12].
3. Modular RAG. Modular RAG differentiates from the Naive RAG framework by offering greater diversity and flexibility in the entire process. It not only integrates various methods to expand functional modules, such as a search module in similarity retrieval but also applying a fine-tuning approach to the retriever [12], [14].

C. Multimodal Retrieval-Augmented Generation

Especially in the context of industrial applications, a significant portion of knowledge is stored in non textual formats and multimodal such as images, structured data (e.g. tables, graphs), audio or video. For example, in semiconductor manufacturing, retrieved diagrams or tabular specifications can provide visual context to support technical question answering tasks [15], [4]. With the rise of multimodal LLMs, expanding their capabilities to process, reason and generate outputs across diverse modalities, further opportunities for RAG systems have emerged [16]. This extensions enables RAG systems to retrieve information and generate output across different modalities [11], [16]. Consequently requiring modality-specific pipelines, including document segmentation, structural parsing for tables and image extraction [17]. However, cross-modal alignment and reasoning introduce new challenges, such as compositional reasoning, retrieval biases, redundant retrieval or biases from retrieved documents [16]. Additionally, recent studies on the performance and limitations of various multimodal RAG architectures to face these challenges have been lacking.

III. RELATED WORK

In the research field of LLMs and RAG inside the semiconductor development process, several studies have explored the potential of LLMs for various use cases. A overview of relevant related work is provided in II in the appendix . It is important to acknowledge that the authors do not assert the table to be comprehensive, but rather offer a curated selection of relevant materials to the best of their knowledge.
Among the approaches that have been listed, DocEDA and D2S-FLOW exhibit the highest degree of conceptual and architectural overlap with the present work. Both utilize multimodal RAG pipelines on semiconductor design related documents. This warrants a more detailed examination and differentiation.

DocEDA. DocEDA is an automated system that utilizes a multimodal RAG architecture for the extraction of electrical parameters and the design of analog circuits. The method combines a range of techniques, including document layout analysis, key component parameter extraction using Chain-of-Thought (CoT), image-to-circuit transformation, and circuit layout optimization. Its EDocNet model has been demonstrated to achieve high levels of accuracy and recall in the classification of document regions. The retrieved parameters and diagrams are optimized by an iterative retrieval approach, processed through CoT-guided extraction and enhanced with an GAM-YOLO model to generate structured netlists. In order to optimize circuit layouts, Bayesian optimization and a space-mapping method is employed during the procedure [18].

D2S-FLOW. D2S-FLOW introduces an enhanced RAG-Pipeline for automated parameter extraction and SPICE model generation. The system incorporates Attention-Guided Document Focusing, Hierarchical Document-Enhanced Retrieval, and Heterogeneous Named Entity Normalization to improve multimodal document understanding. Its primary objectives are to address the challenges of reliably extracting parameters from multimodal datasheets and to integrate them directly into SPICE simulation workflows. D2S-FLOW can primarily be utilized for generating SPICE models for circuit design and showed promising evaluation results by achieving an EM-score of 0.86, an F1-score of 0.92, and an EC-score of 0.96 [19].
In contrast to the approaches of DocEDA and D2S-FLOW, which prioritize the extraction of electrical parameters for the purpose of analog circuit design and simulation, our approach addresses different kind of problems in the use of technical datasheets. These include the complexity of cross-references, the heterogeneity of multimodal content (text, tables, diagrams), and the resulting inefficiencies in manual information retrieval. While both of the compared systems implement multimodal RAG techniques, they are more focused on structured output generation [18], [19]. Our system has been designed to assist various stakeholders by enabling dialog-based access to technical knowledge. The system generates traceable, context-rich, and multimodal answers based on retrieved content, thereby reducing search time and increasing information accessibility across and beyond the design process.

IV. PROPOSED SOLUTION: CHATSHEETPT

The following sections introduce our proposed solution, ChatSheetPT, a specialized multimodal RAG system designed to improve access to complex technical information in design specifications. As outlined in previous sections, these documents are often large, fragmented, and multimodal by combining text, tables, and diagrams, thereby pose significant challenges for traditional search methods. To address these challenges, ChatSheetPT offers a multimodal augmentation pipeline to produce accurate, referencable, and multimodal responses through a dialogue-based interface. The primary objective is to reduce the inefficiencies involved in manually extracting relevant information from datasheets.

A. Requirements

To ensure that the architecture, implementation, and evaluation of the system aligned with the overall objective of improving information accessibility, a set of core requirements was defined at the initiation of the research project. The following system requirements were identified:

- **R1:** The system must be able to process and classify technical PDF documents containing multimodal content such as text, tables, and diagrams.
- **R2:** The system must automatically segment the documents into semantically meaningful chunks, including paragraphs, table structures, and visual references, with corresponding metadata to enable citeable retrieval.
- **R3:** All document chunks must be transformed into semantic vector representations and stored in a retrievable index, allowing similarity-based retrieval.
- **R4:** The retrieval module must be able to return the top-k most relevant document units for a given user query based on semantic alignment.
- **R5:** The retrieved content must be integrated into the user prompt in a way that preserves context relevance and optimally supports the generation process by the LLM.
- **R6:** The system is required to generate context-aware, traceable, and multimodal responses using a LLM, grounded in the retrieved documents and suitable for domain-specific queries.
- **R7:** Generated answers have to include footnotes to the specific document from which the information was derived, ensuring traceability.
- **R8:** The system architecture must allow for modular integration and easy interchangeability of the LLM (e.g. OpenAI, Azure, Anthropic, or open-source models like LLaMA), enabling flexibility.
- **R9:** The system must be capable of explicitly communicating uncertainty. If relevant information cannot be

found in the source documents, the system has to respond accordingly instead of fabricating answers.
- **R10:** The generation pipeline must avoid hallucination by constraining the model to only use retrieved content. Therefore, it has to be part of the evaluation.
- **R11:** The system must ensure answer completeness by aiming to include all relevant technical details from the retrieved content. Partial or fragmentary answers should be avoided unless explicitly requested.

B. Architecture

The proposed architecture follows a modular pipeline reflecting the initial project requirements (see R1–R11). The system is composed of five main stages: indexing, chunking, embedding, retrieval, and generation. The overall design is inspired by the advanced RAG framework proposed by LangChain for semi-structured and multimodal data processing, and it incorporates contextual retrieval strategies introduced by Anthropic [20], [21].

The process begins with the ingestion of PDF-based datasheets. These documents are then sent to a classifier that separates the content into three streams based on modality: text, tables, and images (R1, R2). This separation ensures that each content type can be handled with tailored processing techniques.

During the chunking stage, text is divided into meaningful units and enhanced with contextual information to maintain semantic continuity (R2). Multimodal data as evidence in our case can be challenging for RAG systems for at least two reasons: Initially, the process of text splitting has the potential to disrupt the integrity of tables, thereby corrupting the data during retrieval [20]. Secondly, during embedding tables present challenges related to semantic similarity search, which can result in the loss of information [20]. LangChain outlines three strategies for incorporating visual content into RAG pipelines, particularly when working with multimodal LLMs:

Option 1: Embed both text and images using multimodal embedding models (e.g., CLIP), then pass the raw content to a multimodal LLM for answer synthesis.

Option 2: Generate descriptive summaries for images and tables, embed and retrieve only the summaries, and use them during generation.

Option 3: Generate descriptive summaries with references to the raw image, retrieve them, and pass both the raw image and text to a multimodal LLM for response generation.

We opted for option three because we can leverage the capabilities of multimodal LLMs to process both text and images. For non-textual content, such as tables and diagrams, we consequently designed the system to generate descriptive summaries that are split into semantically aligned chunks (R2, R11). These descriptions are used not only for embedding, but also to enable the cross-modal searchability [20]. Additionally, contextual augmentation techniques are applied to the chunks, which improves retrieval precision without increasing the risk of hallucinations [21].

In the following embedding stage, all chunks are embedded into a shared semantic vector space. Each embedding is stored alongside a unique document ID and metadata to maintain traceability (R3, R7). This approach allows us to synthesize responses during the generation process by using retrieved embeddings while maintaining access to the original content via identifier-based lookups. The raw tabular and image data is persisted in a separate document store, to retrieve the raw content during answer generation. Despite the fact that modern vector databases provide the capability to embed Base64-encoded raw images, storing full binary image strings in a vector store is inefficient and therefore not recommended. Vector stores have been optimized for similarity search on dense vectors, rather than for retrieving raw binary image data. Therefore, we follow the decoupled strategy which employs the multi-vector retriever paradigm by LangChain. This retriever combines a semantic similarity search via a vector store and structured lookups via a document store (R4). This type of retriever is necessary to preserve detail and avoid context loss in the generation process by downstreaming both the embedded representation of the document chunks and the referencing original table or diagram in the context (R5), (R6). Once the relevant chunks have been retrieved, they are sent through a reranking component that prioritizes them based on query alignment. The top-ranked content is forwarded to the generation stage, where a modular, multimodal LLM interface processes the prompt to generate a grounded response (R6, R8). The LLM is configured to answer only based on the retrieved input, ensuring that the generated content remains faithful to the source. Furthermore, when no relevant information is found, the system is designed to explicitly communicate uncertainty to the user instead of providing incorrect information (R9). This protects both the reliability and the transparency of the system.

Finally, the output includes metadata with references to the original document, enabling traceability and verification (R7). The answer generation logic aims for completeness, ensuring that technical details, such as parameter lists or descriptions, are returned in full when present in the source (R11).

A comprehensive visualization of the architecture can be found in Fig 2.

C. Implementation

The implementation of ChatSheetPT is based on the LangChain framework, which offers modular building blocks and comprehensive integrations of state-of-the-art techniques for RAG systems. This makes the system to be more future-proof by incorporating vendor optionality into the infrastructure design. Additionally, the framework has proven to be widely adopted in the industry, resulting in a large community and extensive documentation.

The following subsections outline the key implementation aspects of the system across the four stages: parsing and indexing, retrieval, augmentation, and generation. Each subsection describes the corresponding design and how it aligns with

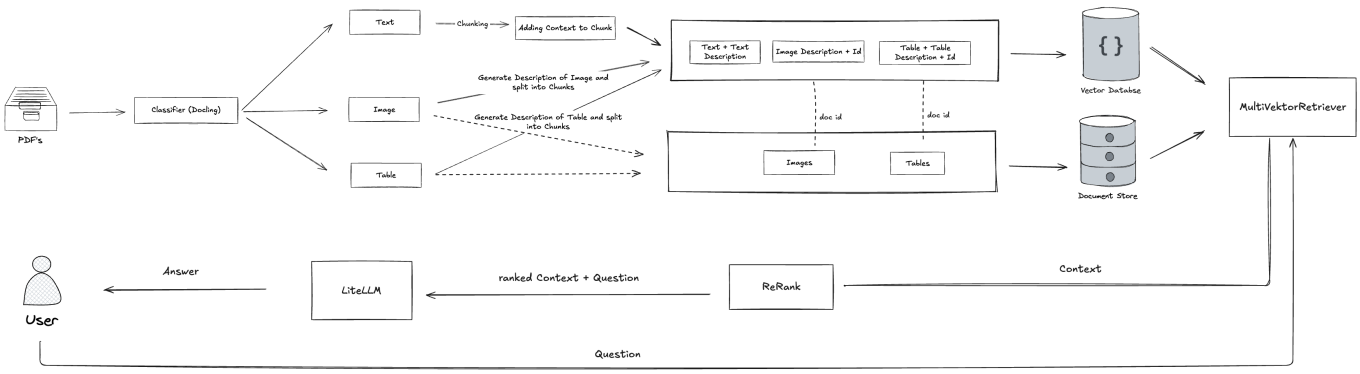

Fig. 2: Architecture of ChatSheetPT

the requirements and architecture introduced in the previous sections.

1) Parsing and Indexing: The pipeline for the RAG system begins by adding information from the datasheets to a knowledge base. Due to the multimodality of the documents, the parsing and indexing stage is crucial step in the overall process, as it ensures that the content is correctly segmented and indexed for later retrieval. Consequently, the quality of the data depends on the accuracy of the parsing and indexing stage. In order to meet the requirements R1 and R2, a comprehensive framework is needed. The interoperability of the LangChain framework allows for the integration of various components, such as PDF parsers, OCR engines, and document classifiers. After extensive research, we identified ”unstructured” as the most suitable PDF parser for our use case. Unstructured is an open-source Python library that offers preservation of document structure and metadata, in contrast to conventional ETL tools that convert documents into plain text. Notably, it offers native support for multimodal content segmentation, including text, tables, and images [22]. Following a series of experiments with various OCR-based and transformer-based models to extract elements from the documents it was found that there were too many incorrect extractions or document element interpretations, including misinterpreted diagrams as tables. Given the suboptimal results obtained, we opted to transition to an alternative that was not open source, by leveraging the Mistral OCR API. We can confirm that the Mistral OCR API provides a high level of accuracy in extracting text and metadata from PDF documents. Given that the Mistral OCR API is not open-source and therefore expensive, we transitioned to our final solution, ”Docling”. Docling is an open-source toolkit for document conversion published by IBM Research in July 2024 [23]. It establishes a unified DoclingDocument data model for rich document metadata and integrates seamlessly with LangChain. A detailed description of Doclings’s pipeline can be found in Fig. 3.

2) Chunking and Contextualizing: After the serialization of the document into the DoclingDocument data model, the content has to be separated into three streams based on modality: text, tables, and images, for individual processing.

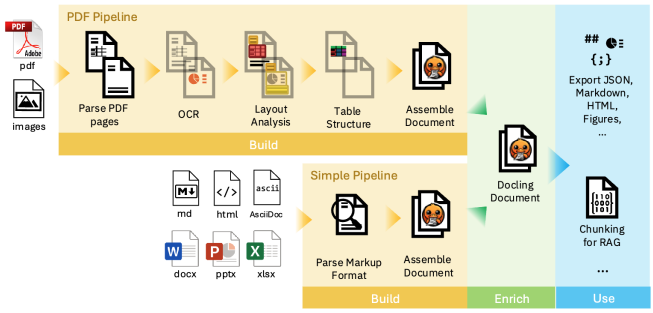

Fig. 3: Sketch of Docling’s pipelines [23]

In order to enhance the quality of the retrieval results, a comprehensive summary was appended to each text, table, and image element [21], [24]. To meet the requirement R2 and R3, another challenge is to find the right chunking strategy. Chunking divides the content into semantically meaningful units, which is essential for effective retrieval and generation [17], [12]. In our case, we opted for a chunking strategy that preserves the semantic meaning of the content, while also ensuring that the chunks are not too large to overwhelm the LLM. More precisely, we used the HybridChunker from Docling, applying tokenization-aware refinements on top of document-based hierarchical chunking. The Chunker performs two passes over the document. The initial pass divides chunks only when necessary, based on the provided tokenizer. The second pass merges chunks only when feasible. For the tokenizer, tiktoken was selected as the Byte-pair encoding.

3) Vectorization: Following the process of chunking and enriching content with context, the next step involves embedding these chunks into a vector space to enable searchability via semantic similarity, in order to address requirements R3 and R4 [12]. This includes also image and table descriptions that were previously generated during the chunking stage and enhanced with metadata, such as a corresponding image or table identifier, content type, and page number.

For the embedding model, we found the `text-embedding-3-large` model from OpenAI to be

sufficient enough for our use case. In terms of infrastructure, two separate components were used for storage:

- A **Vector Database** to store embedding vectors.
- A **Document Database** to store the raw, original data (including text, tables, and images).

For the vector store, we selected `ChromaDB` due to its native support in LangChain, self-deployment capabilities, and its open-source nature. The document store was implemented using a lightweight key-value store `LocalFileStore` provided by LangChain, which supports storing binary data and internally take care of encoding and decoding data for their specific needs.

As previously mentioned, the semantic descriptions of the images are embedded and stored in the vector store, while the actual raw binary image data is stored externally in the document store and referenced only by the unique identifier.

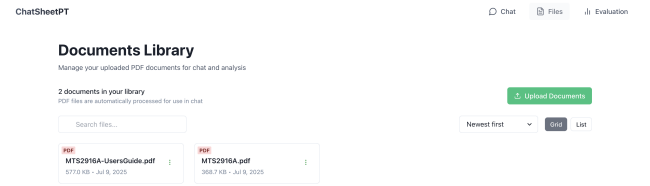

Fig. 4: Datasheet Upload of ChatSheetPT

4) Retrieving: Once the content has been embedded and stored, the subsequent step is to retrieve relevant information in response to user queries. The task of retrieving data is managed by a `MultiVectorRetriever`, a component of LangChain, which has been developed to address the current limitations of conventional retrieval mechanisms in RAG systems. In conventional pipelines, the same document representation is frequently employed for both similarity retrieval and answer synthesis. However, this coupling can result in information loss or redundancy, particularly in the context of multimodal documents where tables, diagrams, and text have different semantic characteristics [20]. The `MultiVectorRetriever` decouples these two processes by distinguishing between short, embedding-optimized chunks for retrieval and full, original content for generation. When a user submits a query, the retriever first performs a similarity search against the vector database (ChromaDB) to identify the top-k most relevant chunks based on their semantic embeddings (R4). The result is a list of semantically relevant chunks, each of which contains metadata including the unique identifier that links it to the original parent document in the document store. These identifiers are then used to queries the document store to retrieve the complete content, such as images and tables but also metadata which is important for citation (R7).

5) Generation: In the final stage of the pipeline, the retrieved chunks have to be transformed into a coherent, contextually grounded response. This phase is critical, as it determines the success of the system in meeting the requirements R5 to R11. To maximize the alignment between the retrieved content and the user query, we first apply a re-ranking step using FlashRank. FlashRank is a lightweight Python library that enables re-ranking to retrieval pipelines and is based on SoTA LLMs and cross-encoders. It supports both Pairwise / Pointwise (cross encoder based) and Listwise re-ranking methods [25]. We chose the Listwise approach due to its more efficient and fast processing. The idea is to avoid any extra overhead. To that end models with really small footprint that doesn't need any specialised hardware and yet offer competitive performance [25]. The concrete model utilized is the `rank-T5-flan` model, which has been labeled as the best non cross-encoder reranker out of the available models.

After the re-ranking step, the top-k chunks are passed to the LLM for answer generation. The LLM is configured to generate responses based solely on the retrieved content, ensuring that the answers are grounded in the source material (R6). This is achieved by using the following prompt:

> *Answer the question based only on the following context, which includes text, tables, and images (if present).*
> Context: {context_text}
> Question: {user_question}

If the retrieving process fails the system is instructed to reply with: *I apologize, but I couldn't find relevant information to answer your question.* (R9).

For the actual generation, we employ LiteLLM as an layer between our system and the LLM providers. LiteLLM abstracts various APIs enabling modularity for the users, as stated in R8. Future integration of alternatives are therefore effortless to implement and does not require structural modifications to the pipeline. To achieve complete satisfaction of R6 and R7, and to ensure both traceability and multimodal outputs, the output is structured as follows:

> "answer": {answer},
> "context": {"images": [*images*], "tables": [*tables*],
> "texts": [*texts*]},
> "retrieval_stats": {source_counts}

D. Evaluation

This section presents the experimental evaluation of our retrieval-augmented pipeline, which has been designed to generate textual responses based upon information from semiconductor design specifications. Given the absence of an adequate baseline system to compare with, our research is oriented towards an elementary analysis to give a first impression of the systems capabilities, limitations and further potential. The objective of this evaluation is not to provide

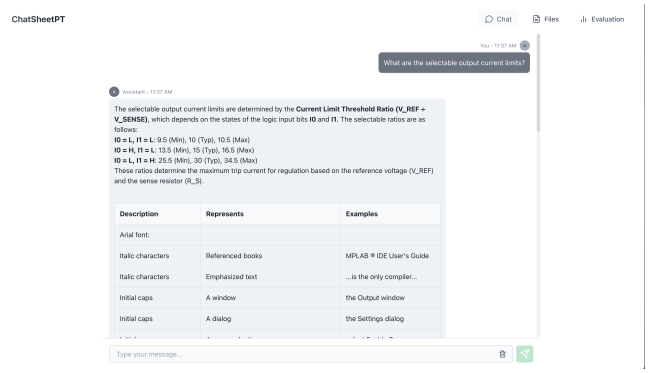

Fig. 5: Chat Interface of ChatSheetPT

statistical significance, but rather to demonstrate functional validity, highlight strengths and weaknesses of the system, and identify areas for improvement. Therefore the evaluation of the system's performance is designed to focus on critical aspects such as correctness, conciseness, hallucination, groundedness, relevance, and helpfulness of the responses. A set of ten domain-specific questions was designed to simulate realistic information needs of different stakeholders. The sample queries and their corresponding answers are documented in Appendix B.

The evaluation was conducted using the LangSmith observability platform alongside the OpenEvals library. The core evaluation methodology follows the LLM-as-a-judge paradigm. In this paradigm, LLM's judge the quality of the generated responses against defined criteria. For the LLM model to judge the responses, we used the `gpt-4.1` model from OpenAI. The following evaluation criteria were selected to reflect the specific requirements of the system (especially R6, R10, and R11):

- **Correctness:** This metric measures how accurately the generated response reflects the expected reference answer.
- **Helpfulness:** It evaluates the extent to which the generated response addresses the user's original question.
- **Groundedness:** It assesses the extent to which the generated output remains faithful to the retrieved context.
- **Retrieval Relevance:** It measures how well the retrieved documents align with the user's query in terms of semantics and context.
- **Hallucination:** It evaluates whether any parts of the generated output are fabricated or cannot be supported by the given input and context.
- **Conciseness:** The goal is to evaluate responses based on precision. Information that is not relevant is graded negatively [6], [12], [26].

As summarized in the Table I, the evaluation results reveal important insights into the system's capabilities and limitations. The metrics for **groundedness** (1.0), **relevance** (1.0), **hallucination** (1.0), and **helpfulness** (1.0) reached perfect scores. These results indicate that the system reliably retrieves semantically relevant content and generates responses exclusively based upon the retrieved documents without hallucinating information (see R6 and R10). The mean **correctness** was slightly lower, at 0.8. An inspection of the responses that had been scored as *incorrect*, revealed that the issues did not stem from factual errors. Rather, it appeared to be due to partially correct answers that lacked specific details that are present in the reference answers. This underscores a current limitation of the system, in its ability to constantly include complete technical details (R11). The lowest score was observed in the **conciseness** metric, with a mean of 0.4, indicating generated responses include verbose or redundant information. This might be a consequence of the system's attempt to ensure completeness, which can lead to overly detailed responses.

The evaluation confirms that ChatSheetPT is capable of generating contextually grounded responses across a small set of realistic questions. The robust performance in hallucination avoidance underscores the effectiveness of RAG. At the same time, the results suggest potential areas for optimization, particularly with regard to precision and completeness. Despite the evaluation is based on a limited dataset, it validates its core functionalities.

V. DISCUSSION

This work presents some valuable insights into the current capabilities and limitations of multimodal RAG systems in the semiconductor industry. However, several architectural, implementation, and evaluation aspects require further critical reflection. The decision to use LangChain as the foundational framework provided modularity enhanced by rapid integration capabilities of components, such as retrievers, vector stores, or embedding models. While this choice enabled faster development, it also kind of limited the system to LangChain-supported tools by making every other integration a workaround. For example, Docling was integrated as a custom component, which required additional effort to ensure compatibility with the LangChain framework. In comparison, Unstructured, which was initially considered, would have provided a more seamless integration and better documentation by being natively integrated. The pipeline's reliance on a single embedding model (`text-embedding-3-large`) offered sufficient results but limited exploration of alternative strategies. However, alternative models were not tested in the current setup, which limits the generalizability of the retrieval's effectiveness. Choosing the `MultiVectorRetriever` was a crucial decision, as it maintained the integrity of multimodal content, but also introduced complexity in managing two separate data stores especially during the indexing phase. For the databases we opted for `ChromaDB` and `LocalFileStore`, which are both sufficient for prototyping. Alternative vector databases were not yet exploited, but could provide advancements like hybrid search. Similarly, the document store is practical for small-scale use but may require reconsideration for production deployment. One of the system's strengths was adding context to the chunking process. By enhancing

Question	Correctness	Helpfulness	Groundedness	Relevance	Hallucination	Conciseness
Q1	0.0	1.0	1.0	1.0	1.0	0.0
Q2	1.0	1.0	1.0	1.0	1.0	0.0
Q3	0.0	1.0	1.0	1.0	1.0	0.0
Q4	1.0	1.0	1.0	1.0	1.0	1.0
Q5	1.0	1.0	1.0	1.0	1.0	1.0
Q6	1.0	1.0	1.0	1.0	1.0	1.0
Q7	1.0	1.0	1.0	1.0	1.0	0.0
Q8	1.0	1.0	1.0	1.0	1.0	1.0
Q9	1.0	1.0	1.0	1.0	1.0	0.0
Q10	1.0	1.0	1.0	1.0	1.0	0.0
Ø	**0.8**	**1.0**	**1.0**	**1.0**	**1.0**	**0.4**

TABLE I: Evaluation Results for Sample Questions

each chunk with summaries and metadata, the system was able to increase retrieval accuracy [21]. This was particularly helpful for the multimodal content like diagrams and tables, which otherwise would not be semantically searchable with our setup. Nevertheless, this approach may have resulted in lengthy responses at the cost of conciseness. It is also notable that while embedding models are good at capturing semantic meaning, they struggle when exact matches are required. Traditional lexical methods, like Best Matching 25 (BM25) can assist in these situations, by applying a ranking function to find precise word or phrase matches [21]. To address retrieval imprecision, we incorporated a re-ranking process using FlashRank. Though not tested in isolation and therefore difficult to estimate its unique contribution, this module is likely to be the reason behind the system's high groundings and relevance scores. Through reordering of the top-ranked results as a function of contextual alignment with the query, reranking enhances the input quality sent to the LLM. For the LLM-as-a-judge evaluation approach, only GPT-4.1 was used as the only judge without any reference evaluation. This may lead to overlooking of subtle differences in semantic that are only visible to a field expert.

VI. LIMITATIONS

While ChatSheetPT is still in an early stage of development, it demonstrates promising results in generating contextually grounded responses. Nevertheless, it is important to acknowledge its current limitations regarding the implementation and evaluation, in order to provide a transparent perspective. Because it has never been tested in a production environment, the system's performance in real-world scenarios remains uncertain. Therefore, the limitations relate to generalizability of the findings and form the basis for future research.

First to consider is the small-scale dataset containing only two datasheets and ten domain-specific questions. Although these questions were supposed to reflect real stakeholder information needs, the small sample size limits the generalizability of the evaluation results on its performance on larger collections of documents. Second, the evaluation was exclusively conducted using a single LLM model (`gpt-4.1`) as the judge. This introduces potential bias and inconsistency, as the evaluation results may not be equal across different LLMs. Furthermore, since no human annotators were involved in the evaluation, no cross-validation by human judgment could have been conducted. Third,the lack of an adequate baseline system to compare with limits the ability to assess the system's performance against existing solutions. This absence makes it difficult to draw meaningful conclusions about the relative performance of the system's architecture in comparison to other approaches. Therefore, making it difficult to claim strengths and areas for further improvements. In addition to these limitations, the system's individual pipeline components where not evaluated in isolation. This means that the individual performance of e.g. the re-ranking, pdf extraction, or contextual enrichment can not be measured. Consequently, it is not possible to determine the exact contribution of each component to the overall system performance. Lastly, the choice of the chosen models for embedding, re-ranking, and generation was not benchmarked. This leaves room for discussion about potential performance of alternative models. Moreover, no fine-tuning of embeddings or generation models was performed, which could have further improved the system's performance and thereby remain unexplored.

In conclusion, since the research demonstrates functional feasibility, the limitations outlined should be considered when interpreting the findings. They can provide a foundation for improvements in future work.

VII. FUTURE WORK

Acknowledging the limitations of the current version of ChatSheetPT, several areas for future work can be identified to further enhance the capabilities and performance. One logical conclusion is to conduct a more thorough evaluation across a broader dataset, which includes a larger number of datasheets and a more diverse set of questions. This would make it possible to gain more reliable insights into the system's performance and generalizability. Additionally, involving human judges alongside LLM-as-a-judge would result in a more reliable validation, especially with subjective judgments such as helpfulness or correctness. To improve the retrieval quality future work could explore hybrid retrieval strategies by integrating lexical search methods, such as BM25, alongside semantic search. As research suggest, combining these two approaches can lead to improved retrieval performance

[21]. To further enhance the performance of the embedding model, fine-tuning it into a domain-specific model could be promising [12], [17]. Examining the usage of multimodal embedding models, such as CLIP by OpenAI, is another possible path towards enhancing cross-modal retrieval capabilities. Furthermore, integrating knowledge graphs to improve the system's understanding of relationships between entities in the datasheets, could be worth experimenting with. Regarding the potential optimization during the generation phase, the current system prompt could be further refined by adding few-shot prompting towards more comprehensive responses. To capitalize of the modular approach of the system, future work should also explore the effects of using alternative LLMs. Lastly, deploying the system in a productive environment and integrating it into existing workflows would provide critical insights into its usability under realistic circumstances. Therefore, real world feedback would be invaluable for identifying further areas of improvements.

VIII. CONCLUSION

This research project introduced **ChatSheetPT**, a specialized multimodal RAG system designed to improve access to complex technical information from design specifications. It delivers context-aware, traceable, and multimodal responses, grounded in retrieved knowledge from parsed datasheets, to answer domain-specific user queries. The system was developed as a response to the formulated research question (**RQ**): *How can a domain-specific retrieval augmented generation system be designed and implemented to improve reliable, traceable, and multimodal access to complex design specifications in semiconductor development?*. To address this question, the project utilized a theoretically grounded architecture with modular implementation upon the LangChain framework. The systems pipeline consists of multimodal document parsing, hybrid-chunking, embedding, contextual-retrieval, and reranked generation stages. This results in a proof-of-concept version that demonstrates advanced RAG techniques adapted to the semiconductor industry. The evaluation, while exploratory in its nature, indicates its capabilities in generating grounded, helpful and hallucination-free responses. These results support the assumption that RAG is a promising and suitable approach for making complex and distributed technical information better accessible for various stakeholders. The current limitations do not undermine its contributions, but rather provide a foundation for future research by highlighting areas for improvement. To this end, ChatSheetPT can be seen as a starting point for future research in the field of applied RAG methods for the semiconductor industry.

ACKNOWLEDGMENT

We acknowledge that artificial intelligence (AI) tools, including ChatGPT, DeepL, and GitHub Copilot, were utilized in the process of creating this paper. These tools were utilized only for LaTeX code editing, language refinement, editing, and grammar enhancements and were not used to generate original content or ideas.

REFERENCES

[1] M.-A. Polosky and E.-J. Garcia, "Microsystem Product Development," Nov. 2007.
[2] M. Cirstea, K. Benkrid, A. Dinu, R. Ghiriti, and D. Petreus, "Digital Electronic System-on-Chip Design: Methodologies, Tools, Evolution, and Trends," *Micromachines*, vol. 15, no. 2, p. 247, Feb. 2024.
[3] W. Fang, M. Li, M. Li, Z. Yan, S. Liu, Z. Xie, and H. Zhang, "AssertLLM: Generating and Evaluating Hardware Verification Assertions from Design Specifications via Multi-LLMs," Nov. 2024.
[4] Y. Pu, Z. He, T. Qiu, H. Wu, and B. Yu, "Customized Retrieval Augmented Generation and Benchmarking for EDA Tool Documentation QA," Jul. 2024.
[5] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large Language Models: A Survey," Mar. 2025.
[6] R. Bommasan and et al., "On the Opportunities and Risks of Foundation Models," https://arxiv.org/pdf/2108.07258.
[7] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen, "A Survey on Multimodal Large Language Models," *National Science Review*, vol. 11, no. 12, p. nwae403, Nov. 2024.
[8] A. Hurst and et al., "GPT-4o System Card," https://arxiv.org/pdf/2410.21276, Aug. 2024.
[9] A. Grattafiori and et al., "The Llama 3 Herd of Models," https://arxiv.org/pdf/2407.21783, Nov. 2024.
[10] A. Rastogi and et al., "Magistral," https://mistral.ai/static/research/magistral.pdf, Jun. 2025.
[11] R. Zhao, H. Chen, W. Wang, F. Jiao, D. Long, C. Qin, B. Ding, X. Guo, M. Li, X. Li, and S. Joty, "Retrieving Multimodal Information for Augmented Generation: A Survey," in *Findings of the Association for Computational Linguistics: EMNLP 2023*. Singapore: Association for Computational Linguistics, 2023, pp. 4736–4756.
[12] Y. Gao and et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 2024.
[13] P. Béchard and O. M. Ayala, "Reducing hallucination in structured outputs via Retrieval-Augmented Generation," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, 2024, pp. 228–238.
[14] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis, L. Zettlemoyer, and S. Yih, "RA-DIT: RETRIEVAL-AUGMENTED DUAL INSTRUC- TION TUNING," 2024.
[15] K. Chang, Y. Wang, H. Ren, M. Wang, S. Liang, Y. Han, H. Li, and X. Li, "ChipGPT: How far are we from natural language hardware design," Jun. 2023.
[16] M. M. Abootorabi, A. Zobeiri, M. Dehghani, M. Mohammadkhani, B. Mohammadi, O. Ghahroodi, M. S. Baghshah, and E. Asgari, "Ask in Any Modality: A Comprehensive Survey on Multimodal Retrieval-Augmented Generation," Feb. 2025.
[17] L. Mei, S. Mo, Z. Yang, and C. Chen, "A Survey of Multimodal Retrieval-Augmented Generation," Mar. 2025.
[18] H. C. Chen, L. Wu, M. Gao, L. Shen, J. Zhong, and Y. Xu, "DocEDA: Automated Extraction and Design of Analog Circuits from Documents with Large Language Model," Nov. 2024.
[19] H. C. Chen, Y. P. Xu, and Y. Zhang, "D2S-FLOW: Automated Parameter Extraction from Datasheets for SPICE Model Generation Using Large Language Models," Jun. 2025.
[20] LangChain, "Multi-Vector Retriever for RAG on tables, text, and images," https://blog.langchain.com/semi-structured-multi-modal-rag/, Oct. 2023.
[21] Anthropic, "Introducing Contextual Retrieval," https://www.anthropic.com/news/contextual-retrieval.
[22] "Unstructured open source - Overview," https://docs.unstructured.io/open-source/, Jul. 2024.
[23] N. Livathinos, C. Auer, M. Lysak, A. Nassar, M. Dolfi, P. Vagenas, C. B. Ramis, M. Omenetti, K. Dinkla, Y. Kim, S. Gupta, R. T. de Lima, V. Weber, L. Morin, I. Meijer, V. Kuropiatnyk, and P. W. J. Staar, "Docling Technical Report," Dec. 2024.
[24] "Contextual Retrieval Appendix II."
[25] P. Damodaran, "FlashRank, lightest and fastest 2nd stage reranker for search pipelines." Dec. 2023.
[26] M. Riedler and S. Langer, "Beyond Text: Optimizing RAG with Multimodal Inputs for Industrial Applications," Oct. 2024.

[27] M. Li, W. Fang, Q. Zhang, and Z. Xie, "SpecLLM: Exploring Generation and Review of VLSI Design Specification with Large Language Model," Jan. 2024.
[28] Y. Lu, S. Liu, Q. Zhang, and Z. Xie, "RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Model," Nov. 2023.
[29] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Chip-Chat: Challenges and Opportunities in Conversational Hardware Design," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, Sep. 2023, pp. 1–6.

APPENDIX

APPENDIX A: Related Work

Aspect	**SpecLLM [27]**	**ChipGPT [15]**	**RTL-LM [28]**	**Chip-Chat [29]**	**AssertLLM [3]**	**DocEDA [18]**	**D2S-FLOW [19]**
Primary goal	Generation and review of architecture specifications	Natural language to hardware logic designs	Generating design RTL with natural language instructions	Automatic Natural Language to Hardware Description Language (HDLs) translation	Convertion of design specifications into functional verification assertions	Extracting and transforming of electrical parameters to refine circuit layouts	Parameter extracting and generating corresponding SPICE models
Pipeline type	LLM enhanced with prompt-engineering	Multi-stage prompting including output management	LLM enhanced with prompt-engineering	LLM enhanced with prompt-engineering	Multimodal RAG-Pipline	Multimodal RAG-Pipline	Multimodal RAG-Pipline
Data source(s)	Architecture specifications provided as a prompt	Natural language System Design Instructions	Natural language RTL Informations	Natural language Specifications	PDF Datasheets	PDF Datasheets	PDF Datasheets
Domain-specific tuning	Prompt tuning	In-context learning	prompt engineering technique named self-planning	*No specific tuning*	RAG and customized Instructions via Prompt	RAG and optimization framework for layout refinement	RAG with Attention-Guided Document Focusing, Hierarchical Document-Enhanced Retrieval, and Heterogeneous Named Entity Normalization
Target group	Spec authors	Electronic design engineers	RTL Design teams	Hardware engineers	Verification engineers	Circuit design teams	Electronic design engineers

TABLE II: Comparison of Related Work

APPENDIX B: SAMPLE QUERIES AND ANSWERS

1) **Question:** What are the selectable output current limits?
   Answer: The selectable output current limits are 0%, 33%, 67%, or 100% of the maximum output current.
2) **Question:** Which stepping operations are possible with the PWM?
   Answer: Full, half, and micro-stepping operations are possible with the PWM current control and logic inputs.
3) **Question:** Is a special power-up sequencing required?
   Answer: No special power-up sequencing is required.
4) **Question:** What does a high and what does a low logic signal level cause?
   Answer: A HIGH logic signal level causes load current to flow from OUTxA to OUTxB. A LOW logic level causes load current to flow from OUTxB to OUTxA.
5) **Question:** What does a thermal protection circuitry do?
   Answer: A thermal protection circuitry turns off all drivers when the junction temperature exceeds a safe operating limit of +170°C (typical).
6) **Question:** When are the power bridge and all outputs disabled?
   Answer: The power bridge and all outputs are disabled if VLOGIC is smaller than 4V.
7) **Question:** What do the typical PCB layout guidelines include?
   Answer: Separate power ground planes, supply decoupling capacitors close to the IC, short connections and use of maximized copper areas to improve thermal dissipation.
8) **Question:** What is the MTS2916A DUAL FULL-BRIDGE STEPPER MOTOR DRIVER?
   Answer: The MTS2916A Dual Full-Bridge Stepper Motor Driver Evaluation Board control circuitry is designed to typically operate from a 6V to 12V logic input (internally regulated down to 5V) and a 10V to 30V VLOAD input.

VLOAD provides power to the motor windings. Test points are generously distributed throughout the evaluation board. This gives the user easy access and visibility, facilitating a better understanding of the MTS2916A operating details.

9) **Question:** Which Power Connections does the MTS2916A use?
Answer: The MTS2916A Dual Full-Bridge Stepper Motor Driver Evaluation Board uses a combination of terminal blocks, test clips and one DC power jack for power connections. Connections are as follows: a) Motor Output Connections: - J2-1(A3), J2-2(A1), J2-3(B1), J2-4(B3), J2-5(TP21) - TP11(A1), TP12(A3), TP13(B1), TP14(B3). b) VLOAD (Motor Supply Power): - J4-1(PGND), J4-2(VLOAD) - TP20(PGND), TP18(VLOAD). WARNING: Do not connect more than 16V to these motor supply connections while Jumper JP2 is installed. c) VLOGIC: - J1-1(VLOGIC), J1-2(AGND) - TP2(VLOGIC), TP5(AGND)

10) **Question:** What are the steps to power the MTS2916A Dual Full-Bridge Stepper Motor Driver Evaluation Board?
Answer: Follow these steps to power-up the board: 1. With the supply turned OFF, connect the power to the logic portion of the evaluation board at J1 with the specified voltage (7 VDC to 12 VDC). The logic portion of the evaluation board will typically draw less than 50 mA. 2. If the user's stepper motor requires a voltage that is compatible with the logic supply voltage and the user's source can handle driving the stepper motor windings, install JP2. DO NOT connect power at J4. If powering up the stepper from an additional supply, DO NOT install JP2 and connect the stepper motor supply to J4. J1 power will still be required for the logic supply. 3. 4. Connect the bipolar stepper windings to J2 per the schematic diagram. Turn ON the power supplies. Power sequencing is not required due to the under- voltage lockout circuitry. 5. Toggle the Mode switch to cycle through the five modes, as indicated by the binary LED count. 6. Press the Run switch once to tell the PIC16F883 to send drive information to the MTS2916A with minimal (1V) VREF. Subsequent Run presses increase VREF by approximately 1V up to 5V maximum. This increases the current regulation threshold. 7. The Hold switch tells the PIC16F883 to command the MTS2916A to hold the motor position. 8. The Direction switch tells the PIC16F883 to command the MTS2916A to change the direction of the motor. 9. The Speed Adjust Potentiometer (R4) varies an analog voltage that is read by the PIC16F883 Analog-to-Digital Converter, and varies the speed accordingly.