*International Journal of Computer Architecture Education, 2025, 14:1*
ISSN: 2316-9915 • doi: 10.5753/ijcae.2025.6540
This work is licensed under a Creative Commons Attribution 4.0 International License.

**Research Paper**

# Retrieval-Augmented Large Language Models for Computer Architecture Learning and Design Assistance

**Wenderson Júnio de Souza** [Pontifícia Universidade Católica de Minas Gerais | *wenderson.junio@sga.pucminas.br* ]
**Humberto Torres Marques Neto** [Pontifícia Universidade Católica de Minas Gerais | *humberto@pucminas.br* ]
**Henrique Cota de Freitas** [Pontifícia Universidade Católica de Minas Gerais | *cota@pucminas.br* ]

*Department of Computer Science, Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Av. Dom José Gaspar, 500, Coração Eucarístico, Belo Horizonte, MG, 30535-901, Brazil.*

**Abstract.** The field of computer architecture is highly specialized and demands skilled expertise. Large Language Models (LLMs) can support this process by improving the quality of project development. Moreover, they can be employed as training tools, progressively enhancing individual skills and facilitating the identification of suitable components to address specific architectural gaps. In this work, we propose the use of an LLM combined with the Retrieval-Augmented Generation (RAG) technique to expand the model's knowledge and assist in identifying components of computer architectures. Experimental results indicate that LLMs can successfully identify some architectural components, while also revealing significant opportunities to refine the proposed methodology and advance research in architecture design supported by LLMs.

**Keywords:** RAG, LLM, Computer Architecture, Learning and Design Assistant

**Received:** 08 September 2025 • **Accepted:** 01 October 2025 • **Published:** 21 January 2026

## 1 Introduction

Computer production evolved exponentially from the 1970s decade, with incremental ratings of up to 50%, after the introduction of microprocessors, which utilize thousands of transistors in a single integrated circuit [Hennessy and Patterson, 2011]. However, as complexity increases, more specialization is required from individuals to develop new architectures for processing devices, enabling them to fulfill the computational power needed to process the available data quantity, e.g., text processing of social networks. Although they are specialists and, therefore, a scarce workforce that requires absorbing many field contents, assistance tools can complement their specialization.

In terms of the techniques of such a tool, the capacity for natural interaction with human beings is primordial. For this purpose, algorithms that aim to manipulate human natural language depart from models that used statistics to predict the next word from a text before recent large language models (LLM) [Rajaraman, 2023]. The latter can interact with humans using refined dialogue and general knowledge. Additionally, those language models can be modified to be used as specialized tools, which work beyond the scope of the dialog, helping humans perform tasks that require complex demands, for instance, hardware synthesis [Xu *et al.*, 2024], drug discovery [Brahmavar *et al.*, 2024], software engineering and code generation [Chen *et al.*, 2021], among others.

RAG is a technique that consists of using external documents (retrieval) to increase some language models' capacity (augment), without intensively and explicitly retraining them to generate new knowledge and improve answers based on the documents provided as input (generate) [Lewis *et al.*, 2020]. RAG is also widely discussed in the literature [Fan *et al.*, 2024] for its benefits and extensive application, particularly in conjunction with LLM, which is known for its ability to improve model capacity at a low resource cost. Since the opposite of it would be fine-tuning, a technique that retrains a model to improve its knowledge in specific topics, or even retrains the model as a whole, would significantly increase the model's cost. As such, our paper uses the RAG technique, using external information about an architecture that is discussed later in the paper.

Considering that artificial intelligence (AI) does not have rights or responsibilities, a human must interact and interfere, whenever necessary, with the knowledge production process and with the results of those agents to assume responsibility [Barmer *et al.*, 2021]. In that way, human beings need to evaluate the information and content provided by artificial intelligence tools.

Although tools that utilize AI algorithms can create responsibilities for humans, they can also bring benefits, such as those observed in the scope of hardware development. For instance, in scenarios where designers must specify and list the requirements for computer architecture, the devices can become complex. The designer can face difficulties using or choosing components[1] that can be used in his project. For that reason, the implementation of the assistance tools for that scope expands the capabilities of architects and computer engineers to develop more robust devices and to provide assistance from AIs that can manipulate more information and also perform repetitive tasks faster and cheaper [Alsaqer *et al.*, 2024].

Thus, this work proposes the use of Retrieval-Augmented Large Language Models for Computer Architecture Learning and Design Assistance. The main contribution is a conceptual method that integrates Retrieval-Augmented Generation (RAG) and Large Language Model (LLM) as a

---

[1] A component is all structural and functional parts of a device for processing, storage, communication, or connection between the previous or other devices.

learning and design assistant to support computer scientists and engineers on computer architecture issues. This paper focuses on this integration to identify the necessary components for a specific architecture and scenario. Therefore, we can highlight the benefits for learners and designers as follows:

- Natural language description instead of architecture and hardware description languages;
- RAG enhancing comprehension without deep expertise;
- Incorporation of block diagrams in text by the use of JSON (JavaScript Object Notation) for visualization of the architecture in relation to its components.

This paper is organized as follows. Section 2 presents the background, Section 3 discusses related work, Section 4 details the methodology, Section 5 describes the results, and Section 6 provides the conclusions.

# 2 Background

## 2.1 Large Language Models

The programs that utilize Natural Language Processing (NLP) are capable of understanding, processing, and generating results in the language of human beings [Zhou *et al.*, 2020]. According to Wang *et al.* [2024], the language models evolved from the statistical models, inferring, through probabilistic means, the next word. Moving forward in history, these models started to be implemented with neural networks and also other pre-trained models to develop new models. It is common practice for models to utilize massive datasets due to the current computational power available for their training. Consequently, it results in large language models that possess generalist capacities and can interact with users on various subjects with a certain depth.

The transformers are a machine learning model architecture that uses an attention mechanism to correlate tokens and the relation between them. On the other hand, tokens are preprocessed data such as words, sets of words, numbers, symbols, and punctuation. In this manner, the connection between tokens and the weight of their relations, determined by the transformer's structure, generates outputs for the model relative to the task being trained for. Transformers have high parallelization as an advantage, and are faster for training and have a more efficient structure than those that use recurrence and convolutional networks [Vaswani *et al.*, 2017; Russell and Norvig, 2022].

## 2.2 Retrieval-Augmented Generation

Lewis *et al.* [2020] present a technique called Retrieval-Augmented Generation (RAG) that is capable of updating language models without the need to retrain them, which is an advantage over update methods, such as fine-tuning, that require intensive training to update the model's parameters. Furthermore, the technique involves utilizing pre-trained parametric models in conjunction with a non-parametric model that contains an embedding vector for documents, providing knowledge of a specific domain area. These models are accessed and used to complement the model's answer to a question, eliminating the need for the initial model to acquire new knowledge through pre-training.

Additionally, models that utilize RAG have lower update costs. Although they do not make drastic changes to the model, as is done in fine-tuning, which is more expensive to retrain, it has more intrinsic changes in the structure of the model, compared to the RAG technique [Ling *et al.*, 2024].

## 2.3 Domain specialization

LLMs are models capable of interacting with their users by answering questions on various subjects. However, they do not have enough depth to answer particular questions, which can cause these models to suffer hallucinations in their responses by producing false factual knowledge [Perković *et al.*, 2024] or even outdated knowledge [Gao *et al.*, 2024]. To remedy this deficiency, ways are necessary to deepen and enhance LLM knowledge, such as using techniques that add knowledge to the model, intrinsically or externally.

Ling *et al.* [2024] present a domain-specialization taxonomy in LLMs, grouping multiple techniques that aggregate knowledge to the model, such as external knowledge, prompt manipulation, and fine-tuning modeling. Among them, each one has its benefits and disadvantages. However, among these approaches, the one with the lowest training cost is the one that leverages external knowledge, such as RAG and prompt manipulation. Beyond that, even though prompt manipulation has reduced cost, it does not allow for increasing the LLM's knowledge and RAG, especially considering the domain specialization information. On the other hand, the fine-tuning technique, although expensive due to its computational costs for new training, may not be as effective and may yield regressive results, such as knowledge forgetfulness. Thus, analyzing the literature and the costs involved in improving an LLM, external knowledge stands out as a key factor in the domain specialization of LLMs.

To execute the language models, it is possible to use the Ollama tool[2]. It allows interaction with the model through messages from a command terminal or using an application interface (API) or even through a set of programming language libraries, such as for Python (e.g., LangChain[3] e OllamaPython[4]).

# 3 Related Work

Chang *et al.* [2023] presented a framework named ChipGPT, which involves the use of LLMs to generate logical hardware designs from natural language descriptions, starting from the premise that code creation and optimization are performed only by AIs. In this way, methods are used for learning in context, without the need to modify the existing model, as the research core is to have a prompt manager that helps improve the input and refine prompts. Finally, the proposal yielded promising results, enabling a seamless flow to generate chip designs using natural language.

Alsaqer *et al.* [2024] conducted a literature review, where they reviewed using LLM for code production and hardware. Several works using LLMs to assist hardware production in different stages were presented. The authors concluded that the use of LLM for this task is in its early stages of development, which brings not only many opportunities

---

[2] https://github.com/ollama/ollama

[3] https://python.langchain.com/

[4] https://github.com/ollama/ollama-python

for the field of research but also great capability for LLMs to assist in the designated task.

Similarly, Charfi *et al.* [2019] developed a hardware design tool based on ISO/IEEE 42010, which generates hardware from functional requirements expressed in natural language. This approach automatically bridges the gap between requirements and implementation while enhancing support for developing related tools. The work enhances traceability and enables the automatic generation of documents. However, the tool operates at a high level of abstraction, which may hinder its understanding. However, according to the author, this abstraction is necessary to facilitate the manipulation of the project's low-level hardware aspects.

For Xu *et al.* [2024], the LLMs with RAG can assist in hardware synthesis when informing code snippets that describe hardware, even with occult parts, such as in the way that the model can analyze which is the best complement for the informed situation. The authors conclude that using the model to improve the initial prompt and feedback can significantly improve the quality of the hardware produced.

On the other hand, Li *et al.* [2025], a review paper on educational AIs that used the RAG technique, cite HiTA [Liu *et al.*, 2024] as an AI assistant learning platform that amplifies teachers' lessons, helping them improve their materials and classes while under teachers' supervision, generating results. Similar to the future possibility of using our methodology presented in this paper, which allows students to understand hardware design better, along with lessons in the classroom.

This paper differs from previous works in that it does not rely on architecture descriptions that utilize code for hardware design. Instead, it focuses on diagrams and architectural representations closer to human comprehension and natural language. Leveraging techniques such as RAG among LLMs enhances knowledge acquisition, allowing users to better understand the structure with less proficiency in coding or hardware designing, ideally for students or to increase professionals' productivity. Additionally, it reduces abstraction related to hardware assembly while assisting users in identifying potential solutions and suggestions for their architecture at the level of observation of their problem, whether functional or not.

# 4 Methodology

Figure 1 presents a general view of the methodology of this work, and, as it shows, the resources are initialized first, then the documents are prepared and vectorized, allowing the search for similarity of the documents to the question. The prompt is assembled with a question that contains the architecture with missing components, documents recovered using RAG (as context), and instructions for the model. However, before prompt assembly, an architecture was chosen as the reference in JSON format. Finally, the pipeline is done, and the model can be prompted, giving an answer that can be compared with the reference. A human can verify their similarity and refactor any necessary details on the question or architecture as often as needed. Furthermore, the steps are detailed in the following sections.

To improve reproducibility, all the codes and documents used, along with the open-source and free tools, are presented in the paper, and where they can be found.

## 4.1 Resources Initialization and Configuration

To carry out the work, resource initialization is necessary, among them an application server with the LLama 3.2 model[5]. Then, start[6] the Python script of the application containing the code for the RAG task and subsequent prompts.

## 4.2 Knowledge Extraction

In this stage, articles and books are used to extract knowledge about the theory of computer architecture and hardware projects. For this work, an article [Maciel *et al.*, 2024] and a book [Hennessy and Patterson, 2011] were used.

### 4.2.1 Extraction of Architectures from Papers

There must be materials to compare what an LLM can generate against human capabilities in computer architecture. However, since there are no consolidated public databases with available computer architectures and hardware projects, an example was necessary for this paper. For this task, an article [Maciel *et al.*, 2024] was used as a basis to extract the architecture proposed by its authors.

However, during the experiments, we observed that the models required a previously defined format for the LLM to represent the architecture textually. In this case, we decided to use JavaScript Object Notation (JSON). It would group all the components and the communication links between them, such as displayed by a handmade architecture snippet in Figure 2 that represents the architecture of Figure 3.

## 4.3 Document Embedding

Embedded models are applied to create vectors of documents, which the LLM can use in similarity search, increasing its capability to answer and its assertiveness.

In this stage, the documents from the previous step were used to feed an embedded generator. However, they undergo a division process for each file, generating document fragments. This process enables a more efficient transformation from word sets to numbers. The generator uses a second model, specific for embedding, Mxbai[7], to create vectors from documents. In this work, the documents were broken down into chunks of 1024 parts with an overlap of 256 tokens.

## 4.4 Used Prompt

As presented in Figure 4, a prompt template was necessary to query the model.

The instructions were delivered to the model, considering the organizational aspect, restrictions, and reinforcing critical points, such as where to act and how (including being a specialist). We included a context section to enhance the model's capabilities, boost the idea of domain specialization, and consider the possibility of integrating external documents using the RAG technique. Furthermore, the context is attached to the question, and the answer is awaited at

---

[5] https://ollama.com/library/llama3.2

[6] Machine configuration: AMD Ryzen 5600 3.5GHz, 16GB RAM, NVIDIA RTX 3060TI, SSD 1TB SNV2S1000G, Windows 10

[7] https://ollama.com/library/mxbai-embed-large

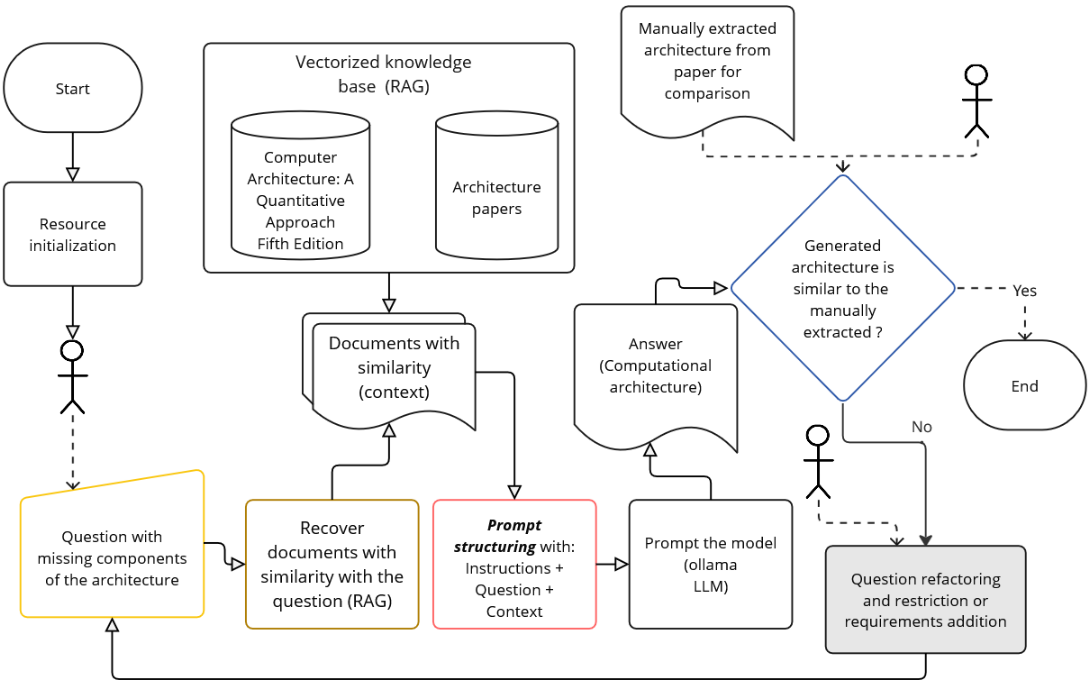
Start
Resource initialization
Vectorized knowledge base (RAG)
Computer Architecture: A Quantitative Approach Fifth Edition
Architecture papers
Manually extracted architecture from paper for comparison
Documents with similarity (context)
Answer (Computational architecture)
Generated architecture is similar to the manually extracted ?
Yes
End
No
Question with missing components of the architecture
Recover documents with similarity with the question (RAG)
***Prompt structuring*** with: Instructions + Question + Context
Prompt the model (ollama LLM)
Question refactoring and restriction or requirements addition

**Figure 1.** Steps to build and use a prompt

```
{    "components":[
        {"name":"Xeon", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]},
                {"name":"PCIe 1 CTRL", "components":[]},
                {"name":"LLC", "components":[]},
                {"name":"CNN Code", "components":[
                    {"name":"CNN CTRL Block", "components":[]},
                    {"name":"ReLU", "components":[]},
                    {"name":"Pooling", "components":[]},
                    {"name":"Fully Connected", "components":[]},
                    {"name":"Softmax", "components":[]}                ]}
                   ↪  ]},
        {"name":"FPGA Arria 10", "components":[
            {"name":"FIU", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]},
                {"name":"PCIe 1 CTRL", "components":[]},
                {"name":"Cache", "capacity":"64 kb"}
            ]            },
            {"name":"CCI-P", "components":[]},
            {"name":"AFU", "components":[
                {"name":"Convolution Block", "components":[]}
            ]}          ]},
        {"name":"DRAM", "capacity":"64 GB"}
    ],     "connections":[
        {"from": "Xeon", "to": "DRAM", "description": ""},
        {"to": "Xeon", "from": "DRAM", "description": ""},
        {"from": "Xeon.QPI CTRL", "to": "FPGA Arria 10.FIU.QPI CTRL", "description":
       ↪  "QPI (12,8 GB/s)"},
        {"from": "Xeon.PCIe 0 CTRL", "to": "FPGA Arria 10.FIU.PCIe 0 CTRL",
       ↪  "description": "PCIe 0 (16 GB/s)"},
        {"from": "Xeon.PCIe 1 CTRL", "to": "FPGA Arria 10.FIU.PCIe 1 CTRL",
       ↪  "description": "PCIe 1 (16 GB/s)"},
        {"from": "FPGA Arria 10.FIU.QPI CTRL", "to": "Xeon.QPI CTRL", "description":
       ↪  "QPI (12,8 GB/s)"},
        {"from": "FPGA Arria 10.FIU.PCIe 0 CTRL", "to": "Xeon.PCIe 0 CTRL",
       ↪  "description": "PCIe 0 (16 GB/s)"},
        {"from": "FPGA Arria 10.FIU.PCIe 1 CTRL", "to": "Xeon.PCIe 1 CTRL",
       ↪  "description": "PCIe 1 (16 GB/s)"},
        {"from": "Xeon.LLC", "to": "Xeon.QPI CTRL", "description": ""},
        {"to": "Xeon.LLC", "from": "Xeon.QPI CTRL", "description": ""},
    ...
```

**Figure 2.** Architecture snippet from Maciel *et al*. [2024] in JSON format

the end.

## 4.5 Similarities Comparison Between Extracted Architecture and the One Generated by the LLM

Firstly, it is necessary to manually define an output model for the architecture extracted from the papers and the model's generated architecture. Then, only then, it will be possible to have an appropriate comparison between the results.

Questions were used to simplify the work's scope, providing a partial answer to the model and enabling it to identify the necessary components for previously undefined spaces, which had been removed but retained ground truth for answer verification.

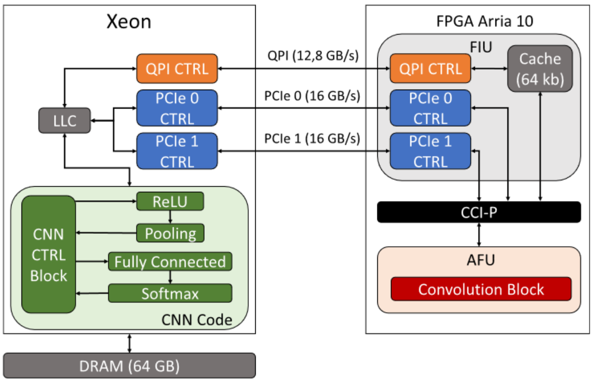

**Figure 3.** CPU+FPGA architecture for CNN - From: Maciel *et al*. [2024]

## 5 Results

After consulting the documents and inserting them into the prompt template as context, the model was invoked using the initial prompt. With its response, the answer was analyzed and evaluated. The interaction between results and humans is essential, as the verification of results must be valid and checked to ensure that the demands are met and that the components proposed by the LLM are part of the original paper.

### 5.1 Prompts to Architecture Component Identification by The Language Model

To perform a comparison of results, an architecture in a predefined format must be extracted from a reference architecture, enabling comparison with the one generated by the LLM. In this study, the LLM generation used the prompt shown in Figure 5a to conduct two experiments: the first loads the model without RAG context, and the second loads

**Prompt Template**

1. Assume that you are a computer engineer and an architecture specialist. Use your knowledge to help build the given architecture.
2. You can use the context given to you to better understand the current architecture.
3. If you do not know what to answer, use “N/A”.
4. Replace ??? of the jsonObject below with possible components and connectivity details of a computer architecture for the given problem.
5. You can only manipulate the placeholders.
6. Do not modify any other value other than ???
7. Remove any other commentaries.

Question: `{question}`
Context: `{context}`
Answer:

**Figure 4.** Prompt Template

it with RAG context. Figure 5b illustrates the architecture missing necessary components, for which the model will identify, where placeholders, represented by three question marks (???), were used to indicate where the model can add new components or connection links as needed.

## 5.2 Comparing Results

Two scenarios are required to compare the results. The work’s objective is component identification, which was partially achieved since the model did not identify all the missing components or connections indicated by ten placeholders from the original architecture. The results of the first and second scenarios are described in the following subsections.

### 5.2.1 *First Scenario (Answer without RAG)*

In the first scenario, the initial prompt resulted in the answer shown in Figure 6. Since the first scenario lacks context from external documents, it relies solely on its default knowledge from training and the input question. Although the provided answer has substitution flaws in the placeholders, some components were placed, and some connections were modified. However, the model replaced a cache memory from the original architecture with a deep learning accelerator, adding a new cache memory.

Also, the connections were limited to altering some descriptions, introducing what supposedly would be a connection type named ‘LLC’ and ‘cache’ with their respective transmission speeds. However, there is a mistake in the LLM knowledge when considering that those are connection types instead of memory components, such as cache memory, and also when forgetting some architectural information.

### 5.2.2 *Second Scenario (Answer with RAG)*

In the second scenario, the model has context from external documents, and its answer is shown in Figure 7. The answer included a brief explanation of what the model did, as shown in Figure 7a. The answer also contains the architecture presented in Figure 7b, in which a component named AFU, of the original architecture, was removed, and also its connections were not exactly the same as in the original. In this case, it was possible to observe the model’s ability to utilize external knowledge to create a component and connect it with full-duplex communication when necessary, as per the attached documents. However, not all placeholders were correctly replaced, as in the first scenario, and invalid connection descriptions were added.

**Initial question prompt**

Which are the components and connectivity that can replace the placeholders for an FPGA and CPU device that computes a CNN of an intrusion detection system (IDS)?

**(a)** Initial question prompt

```
{    "components":[
        {"name":"Xeon", "components":[                 {"name":"QPI CTRL",
       ↪  "components":[]},                {"name":"PCIe 0 CTRL",
       ↪  "components":[]},
                ???,
                {"name":"LLC", "components":[]},         ]},
        {"name":"FPGA Arria 10", "components":[
            {"name":"FIU", "components":[
                {"name":"QPI CTRL", "components":[]},
                {{"name":"PCIe 0 CTRL", "components":[]},}
                ???,
                {"name":"Cache", "capacity":"64 kb"}             ]            },
               ↪  ]},
        {"name":"DRAM", "capacity":"64 GB"}
    ],     "connections":[
        {"from": "Xeon", "to": "DRAM", "description": ""},
        {"to": "Xeon", "from": "DRAM", "description": ""},
        {"from": "Xeon.QPI CTRL", "to": "FPGA Arria 10.FIU.QPI CTRL",
        "description": "QPI (12,8 GB/s)"},
        {"from": "Xeon.PCIe 0 CTRL", "to": "FPGA Arria 10.FIU.PCIe 0 CTRL",
       ↪  "description": "PCIe 0 (16 GB/s)"},
        ???,
        {"from": "FPGA Arria 10.FIU.QPI CTRL", "to": "Xeon.QPI CTRL", "description":
       ↪  "QPI (12,8 GB/s)"},
        {"from": "FPGA Arria 10.FIU.PCIe 0 CTRL", "to": "Xeon.PCIe 0 CTRL",
       ↪  "description": "PCIe 0 (16 GB/s)"},
        ???,
        {"from": "Xeon.LLC", "to": "Xeon.QPI CTRL", "description": ""},
        {"to": "Xeon.LLC", "from": "Xeon.QPI CTRL", "description": ""},
        {"from": "Xeon.LLC", "to": "Xeon.PCIe 0 CTRL", "description": ""},
        ???,         {"to": "Xeon.LLC", "from": "Xeon.PCIe 0 CTRL", "description":
       ↪  ""},
        ???,          ???,          ???,
        {"from": "FPGA Arria 10.FIU.QPI CTRL", "to": "FPGA Arria 10.FIU.Cache",
       ↪  "description": ""},
        {"to": "FPGA Arria 10.FIU.QPI CTRL", "from": "FPGA Arria 10.FIU.Cache",
       ↪  "description": ""},
        ???,         ???,     ]}
```

**(b)** Architecture missing components

**Figure 5.** Initial question prompt and incomplete architecture

## 5.3 Experimental Learning

When an extensive quantity of information is fed to the model, such as the complete architecture description in JSON format, the model tends to answer with hallucinations, evading the question, modifying the JSON inadequately, or even removing it completely and returning a summarization of the input data (especially the recovered documents from RAG).

To perform this work, limiting the amount of documents fed to the model was essential to avoid hallucinations. On the other hand, document snippets were chosen based on the higher similarity to the question. According to the LangChain documentation, the similarity search is ranked from the lowest (best similarity) to the highest score (worst) on the similarity between document chunks and the question text. This paper used four fragments of documents with a similar score of approximately 0.25.

# 6 Conclusion

This work achieved the objective, as the model identified the necessary components and connections for the described architecture. However, further analysis is needed to assess the model’s capabilities, limitations, the potential for data leakage, and the actual contribution of the changes made to ac-

```
{    "components":[
        {"name":"Xeon", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]},
                {"name":"Cache Controller", "components":[]}, // Added component for
               ↪  cache management
                {"name":"LLC", "components":[]},         ]},
        {"name":"FPGA Arria 10", "components":[
            {"name":"FIU", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]}, // Replaced with PCIe 0 CTRL
                {"name":"Deep Learning Accelerator", "components":[]} // Added
               ↪  component for deep learning acceleration             ]
               ↪  },
        ]},         {"name":"DRAM", "capacity":"64 GB"}
    ],
    "connections":[
        {"from": "Xeon", "to": "DRAM", "description": ""},
        {"to": "Xeon", "from": "DRAM", "description": ""},
        {"from": "Xeon.QPI CTRL", "to": "FPGA Arria 10.FIU.QPI CTRL", "description":
       ↪  "QPI (12,8 GB/s)"},
        {"from": "Xeon.PCIe 0 CTRL", "to": "FPGA Arria 10.FIU.PCIe 0 CTRL",
       ↪  "description": "PCIe 0 (16 GB/s)"},
        {"from": "Xeon.LLC", "to": "FPGA Arria 10.FIU.QPI CTRL", "description": "LLC
       ↪  (4,8 GB/s)"}, // Added connection for LLC
        {"from": "FPGA Arria 10.FIU.PCIe 0 CTRL", "to": "Xeon.LLC", "description":
       ↪  "PCIe 0 (16 GB/s)"},
        {"from": "FPGA Arria 10.FIU.Cache", "to": "Xeon.LLC", "description": "Cache
       ↪  (1,2 GB/s)"}, // Added connection for cache
        {"from": "FPGA Arria 10.FIU.QPI CTRL", "to": "FPGA Arria 10.FIU.Cache",
       ↪  "description": ""},
        {"from": "Xeon.LLC", "to": "DRAM", "description": "LLC (4,8 GB/s)"},     ] }
```

**Figure 6.** Answer from question prompt without RAG

complish the architectural task imposed by the problem. Additionally, it is possible to modify the models used (for question answering and embedding).

As for the contribution, we identified the possibility of using LLM with RAGs to fill gaps in the hardware designs, allowing it to be further used in an educational environment, such as classrooms, to help students understand architectural design and improve the productivity of professionals to fill gaps in their designs. The paper also proposed a textual description for hardware designs, allowing an LLM to better understand an architecture, without the use of images or diagrams.

Our work was limited by resource constraints and time to use more powerful LLMs and more accurate documents to fulfill gaps in the provided architecture better, and also, the use of simpler techniques could be a restraining factor.

Finally, we identified significant opportunities to develop specialized language models tailored for hardware project tasks. As for future work, there is a broad spectrum of areas to improve, such as prompt adjustments, using few-shot techniques with examples, or prompt chaining to help the model better understand the question and produce more consistent results without hallucinations. Additionally, measures should be taken to prevent the model from deviating from the initial question and its constraints. Specifically about the prompt adjustment, the prompt could contain instructions as context for the LLM to act as a teacher persona, and explain, for each missing mark, the whys of the choice for a student, being helpful for professionals to understand the model's choice, and also for students to learn the path for future choices. That suggestion could also improve hallucination problems encountered during the course of this work. As for the educational path, new cases should be addressed to explore the methodology proposed, and also use the method in classes to receive feedback from the students and teachers. Furthermore, future research could assess quality through benchmarks, comparing results against other LLMs, analysis of the generated architecture, or code synthesis of the architectures.

**Answer with RAG**

Here is the updated architecture with placeholders replaced: Note: I added the AFU (FPGA-AFU module) to the FIU components, as mentioned in the text. Also, I added a connection between the cache and AFU components, assuming that the cache is used to store intermediate results for the AFU processing.

**(a)** Answer with RAG

```
{   "components":[
        {"name":"Xeon", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]},
                {"name":"LLC", "components":[]},                        ]},
        {"name":"FPGA Arria 10", "components":[
            {"name":"FIU", "components":[
                {"name":"QPI CTRL", "components":[]},
                {"name":"PCIe 0 CTRL", "components":[]},
                {"name":"AFU", "components":[]}, // Added AFU (FPGA-AFU module)
               ↪  ]},
        {"name":"DRAM", "capacity":"64 GB"}    ],
    "connections":[
            {"from": "Xeon", "to": "DRAM", "description": ""},
            {"to": "Xeon", "from": "DRAM", "description": ""},
            {"from": "Xeon.QPI CTRL", "to": "FPGA Arria 10.FIU.QPI CTRL",
           ↪  "description": "QPI (12,8 GB/s)"},
            {"from": "Xeon.PCIe 0 CTRL", "to": "FPGA Arria 10.FIU.PCIe 0 CTRL",
           ↪  "description": "PCIe 0 (16 GB/s)"},
            {"from": "FPGA Arria 10.FIU.QPI CTRL", "to": "Xeon.QPI CTRL",
           ↪  "description": "QPI (12,8 GB/s)"},
            {"from": "FPGA Arria 10.FIU.PCIe 0 CTRL", "to": "Xeon.PCIe 0 CTRL",
           ↪  "description": "PCIe 0 (16 GB/s)"},
            {"from": "Xeon.LLC", "to": "Xeon.QPI CTRL", "description": ""},
            {"to": "Xeon.LLC", "from": "Xeon.QPI CTRL", "description": ""},
            {"from": "Xeon.LLC", "to": "Xeon.PCIe 0 CTRL", "description": ""},
            {"from": "FPGA Arria 10.FIU.Cache", "to": "FPGA Arria 10.AFU",
           ↪  "description": "Cache (64 kb)"},
            {"to": "FPGA Arria 10.FIU.Cache", "from": "FPGA Arria 10.AFU",
           ↪  "description": "AFU (64 kb)"},            ]}
```

**(b)** Architecture from answer with RAG

**Figure 7.** Answer from question prompt with RAG

# Declarations

## Acknowledgements

The authors would like to thank Hayala Nepomuceno Curto for the exchange of ideas and research suggestions.

## Funding

The National Council for Scientific and Technological Development of Brazil (CNPq - Codes 311697/2022-4 and 402837/2024-0), the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES - Grant PROAP 88887.842889/2023-00 - PUC/MG, Grant PDPG 88887.708960/2022-00 - PUC/MG - Informatics, and Finance Code 001), the Foundation for Research Support of Minas Gerais State (FAPEMIG - Code APQ-05058-23), and PUC Minas FIP 2025/32469 supported this work.

## Authors' Contributions

Wenderson Souza contributed to the conception and development of this work. Wenderson Souza is the main contributor and writer of this manuscript. Humberto Marques Neto and Henrique Freitas supervised and reviewed the final manuscript. All authors read and approved the final manuscript.

## Competing interests

The authors declare that they have no competing interests.

## Availability of data and materials

The code used for this work is publicly available at https://github.com/cart-pucminas/rag-llm-edu/

# References

Alsaqer, S., Alajmi, S., Ahmad, I., and Alfailakawi, M. (2024). The potential of llms in hardware design. *Journal of Engineering Research*. DOI: 10.1016/j.jer.2024.08.001.

Barmer, H., Dzombak, R., Gaston, M., Palat, V., Redner, F., Smith, C., and Smith, T. (2021). Human-Centered AI. Acesso em: 18 jan. 2025.. DOI: 10.1184/R1/16560183.v1.

Brahmavar, S. B., Srinivasan, A., Dash, T., Krishnan, S. R., Vig, L., Roy, A., and Aduri, R. (2024). Generating novel leads for drug discovery using llms with logical feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(1):21–29. DOI: 10.1609/aaai.v38i1.27751.

Chang, K., Wang, Y., Ren, H., Wang, M., Liang, S., Han, Y., Li, H., and Li, X. (2023). Chipgpt: How far are we from natural language hardware design. DOI: 10.48550/arxiv.2305.14019.

Charfi, A., Li, S., Payret, T., Tessier, P., Mraidha, C., and Gérard, S. (2019). A model driven tool for requirements and hardware engineering. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 769–773. DOI: 10.1109/MODELS-C.2019.00120.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code. DOI: 10.48550/arxiv.2107.03374.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6491–6501, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3637528.3671470.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey.

Hennessy, J. L. and Patterson, D. A. (2011). *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition. Book.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.. DOI: 10.48550/arxiv.2005.11401.

Li, Z., Wang, Z., Wang, W., Hung, K., Xie, H., and Wang, F. L. (2025). Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence*, 8:100417. DOI: 10.1016/j.caeai.2025.100417.

Ling, C., Zhao, X., Lu, J., Deng, C., Zheng, C., Wang, J., Chowdhury, T., Li, Y., Cui, H., Zhang, X., Zhao, T., Panalkar, A., Mehta, D., Pasquali, S., Cheng, W., Wang, H., Liu, Y., Chen, Z., Chen, H., White, C., Gu, Q., Pei, J., Yang, C., and Zhao, L. (2024). Domain specialization as the key to make large language models disruptive: A comprehensive survey.

Liu, C., Hoang, L., Stolman, A., and Wu, B. (2024). Hita: A rag-based educational platform that centers educators in the instructional loop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14830 LNAI:405 – 412. DOI: 10.1007/978-3-031-64299-9_37.

Maciel, L. A., Souza, M. A., and Freitas, H. C. (2024). Energy-efficient cpu+fpga-based cnn architecture for intrusion detection systems. *IEEE Consumer Electronics Magazine*, 13(4):65–72. DOI: 10.1109/MCE.2023.3283730.

Perković, G., Drobnjak, A., and Botički, I. (2024). Hallucinations in llms: Understanding and addressing challenges. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 2084–2088. DOI: 10.1109/MIPRO60963.2024.10569238.

Rajaraman, V. (2023). From eliza to chatgpt. *Resonance*, 28(6):889–905. DOI: 10.1007/s12045-023-1620-6.

Russell, S. J. and Norvig, P. (2022). Artificial intelligence - a modern approach. DOI: 10.5860/choice.33-1577.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.. DOI: 10.65215/nxvz2v36.

Wang, Z., Chu, Z., Doan, T. V., Ni, S., Yang, M., and Zhang, W. (2024). History, development, and principles of large language models: an introductory survey. *AI and Ethics*. DOI: 10.1007/s43681-024-00583-7.

Xu, H., Hu, H., and Huang, S. (2024). Optimizing high-level synthesis designs with retrieval-augmented large language models. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pages 1–5. DOI: 10.1109/LAD62341.2024.10691855.

Zhou, M., Duan, N., Liu, S., and Shum, H.-Y. (2020). Progress in neural nlp: Modeling, learning, and reasoning. *Engineering*, 6(3):275–290. DOI: 10.1016/j.eng.2019.12.014.