PrimisAI

HOT CHIPS

HotChips 2024 Tutorial

Hans Bouwmeester, PrimisAI

Outline

1. Introduction to LLMs
2. Introduction to RAG
3. From LLM+RAG to EDA AI-Agent

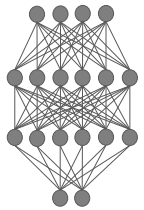
PrimisAI

HOT CHIPS

1. Introduction to LLMs

What is an LLM?

Large Language Model



Architecture

Neural network
Layers of weights
Functions operate on weights
Just a few 100 lines of code!

Parameters (weights)

Datafile with value for each weight
From ~7 Billion to >1 Trillion

Open vs. Closed

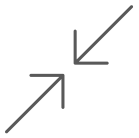
Y/N public availability of:
- Architecture
- Architecture + Weights

Training

Weights are adjusted using training data
For example, Meta Llama 3.1:
- >10 trillion words in training set
- ~60 days training time
- >20000 Nvidia H100 GPUs
- ~ $30M compute costs (assuming H100 ~= $1/hr)

Inference

Weights are used to compute response



Quantization

Compress weights
Size vs. accuracy tradeoff
For example, Meta Llama-70B:
- 70 billion float16 → 140GB file
- 4-bits quantized → 35GB file

Example LLM use for code generation

Hans Bouwmeester

I need a complete UVM testbench for a UART

8/16/2024, 1

RapidGPT

Certainly! I'd be happy to help you create a complete UVM testbench for a UART (Universal Asynchronous Receiver/Transmitter). Before we proceed, I'd like to ask a few questions to ensure I provide you with the most appropriate and tailored testbench:

1. What specific features of the UART do you want to test? (e.g., data transmission, reception, baud rate settings, parity checking, etc.)
2. Do you have a specific UART design (RTL) that this testbench will be verifying, or should I assume a generic UART implementation?
3. Are there any particular test scenarios or corner cases you want to ensure are covered?
4. Do you have any preferences for the configuration of the UART (e.g., data bits, stop bits, parity)?
5. Are there any specific performance metrics or coverage goals you want to achieve with this testbench?

Once you provide this information, I'll be able to create a more tailored and comprehensive UVM testbench for your UART design. The testbench will include components such as the test class, environment, agent(s), sequencer, driver, monitor, and scoreboard, all structured according to UVM methodology.

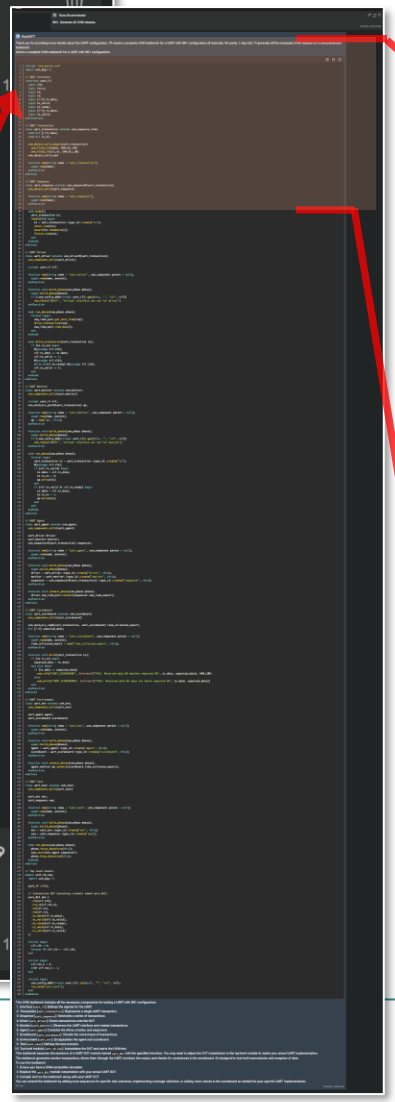
15.8 Sec

8/16/2024, 10:22:34 AM

Hans Bouwmeester

8N1. Generate all UVM classes.

8/16/2024, 1



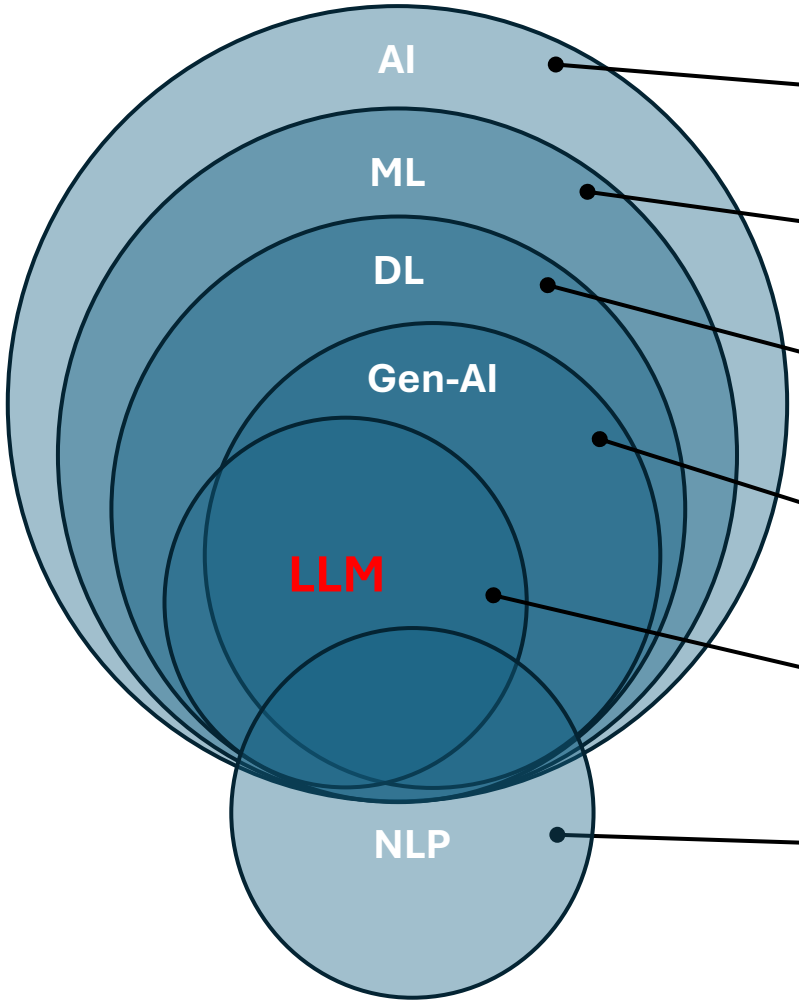
RapidGPT

Thank you for providing more details about the UART configuration. I'll create a complete UVM testbench for a UART with 8N1 configuration (8 data bits, No parity, 1 stop bit). I'll generate all the necessary UVM classes for a comprehensive testbench.
Here's a complete UVM testbench for a UART with 8N1 configuration:

```
include "uvm_macros.svh"
import uvm_pkg::*;

// UART Interface
interface uart_if;
  logic clk;
  logic rst_n;
  logic tx;
  logic rx;
  logic [7:0] tx_data;
  logic tx_valid;
  logic tx_ready;
  logic [7:0] rx_data;
  logic rx_valid;
endinterface

// UART Transaction
class uart_transaction extends uvm_sequence_item;
  rand bit [7:0] data;
  rand bit is_tx;

  uvm_object_utils_begin(uart_transaction)
    uvm_field_int(data, UVM_ALL_ON)
    uvm_field_int(is_tx, UVM_ALL_ON)
  uvm_object_utils_end

  function new(string name = "uart_transaction");
```

PrimisAI

Where do LLMs fit in?



Artificial Intelligence
Reason, learn and act autonomously

Machine Learning
Train on input data, predict on unseen data

Deep Learning
Build model using a neural network

Generative-AI
Generate new data, similar to data used for training

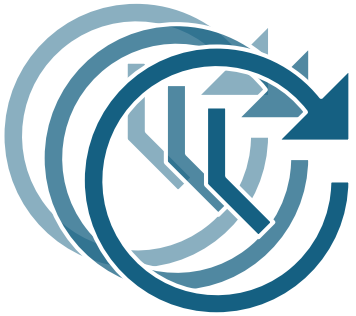
Large Language Model
A model that can perform (N)LP tasks

Natural Language Processing
Interpret natural language

LLM History

NLP → LLM

1966	ELIZA	1st NLP model, rule-based (MIT)
1972	RNN	Recurrent Neural Network (NN with feedback loop and internal state)
1997	LSTM	Long Short-Term Memory (RNN selectively retains past information → attention mechanism)
2017	Transformer	*"Attention is all you need"* (non recurrent, process entire sequence simultaneously) (Google DeepMind)
2018	GPT-1	GPT-1 – 117M par. (OpenAI)
2018	BERT	BERT – 340M par. Bidirectional Encoder Representations for Transformers
2019	GPT-2	2.5B par.
2020	GPT-3	175B par.
2021	Codex	Code Generation (GPT-3-based) → GitHub Copilot
2022	GPT-3.5	GPT-3.5 – 175B par.
2023	GPT-4	1.7T par., Mixture of Experts (MoE, multi-modal)
2024	Llama v3.1	Llama v3.1 – up to 405B par. Open-Source. (Meta)



LLM Use Cases in EDA

IP catalog

Product documentation

Onboarding support

Training

Architecture

Coding & debug

IP integration

Design verification

Documentation

Code commenting

Status summaries

Logfile analysis

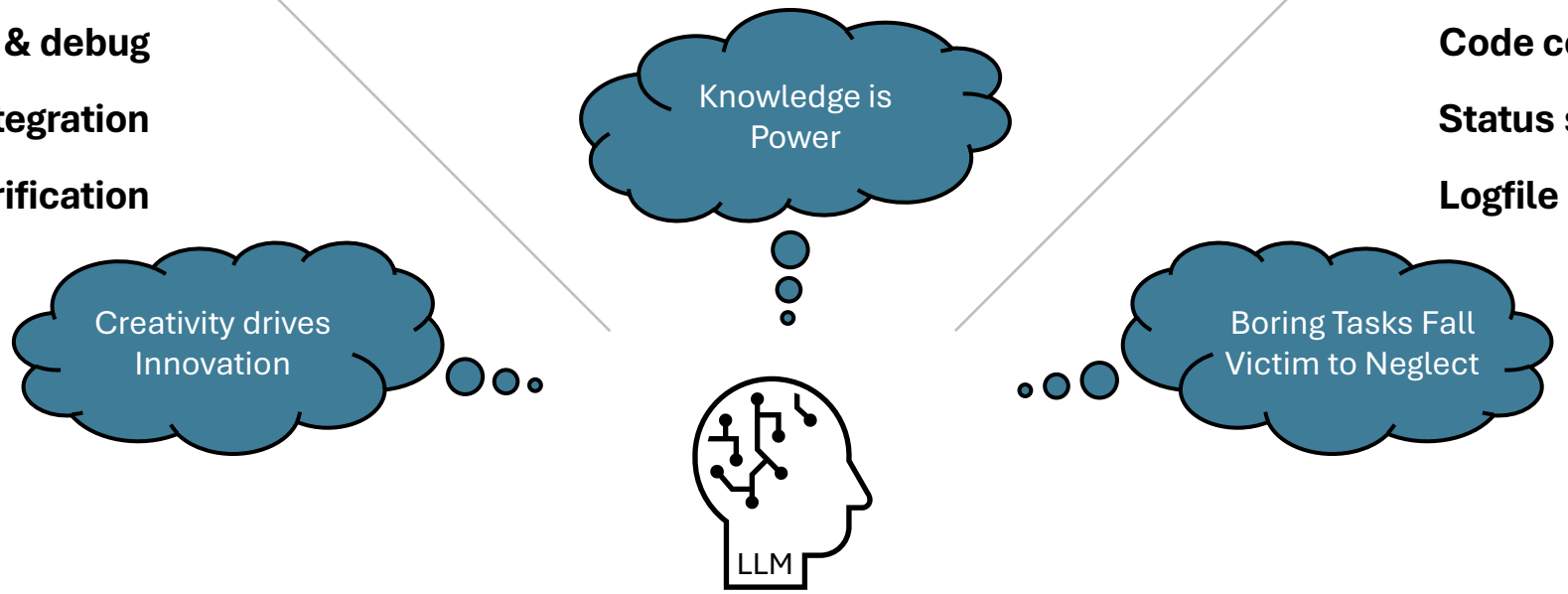
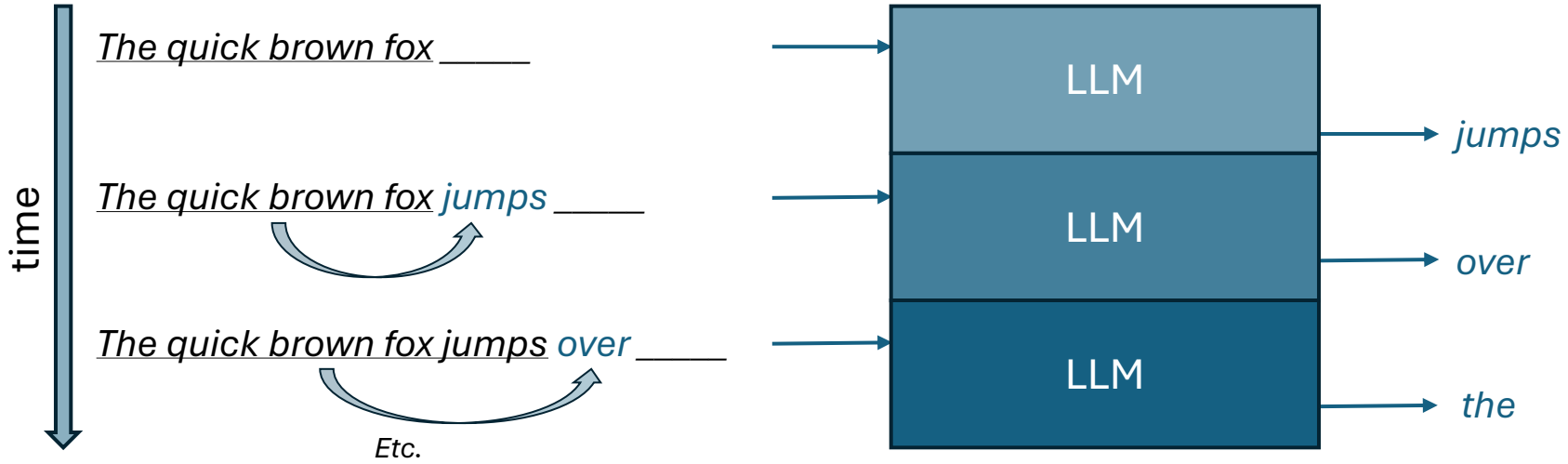


Image: PrimisAI

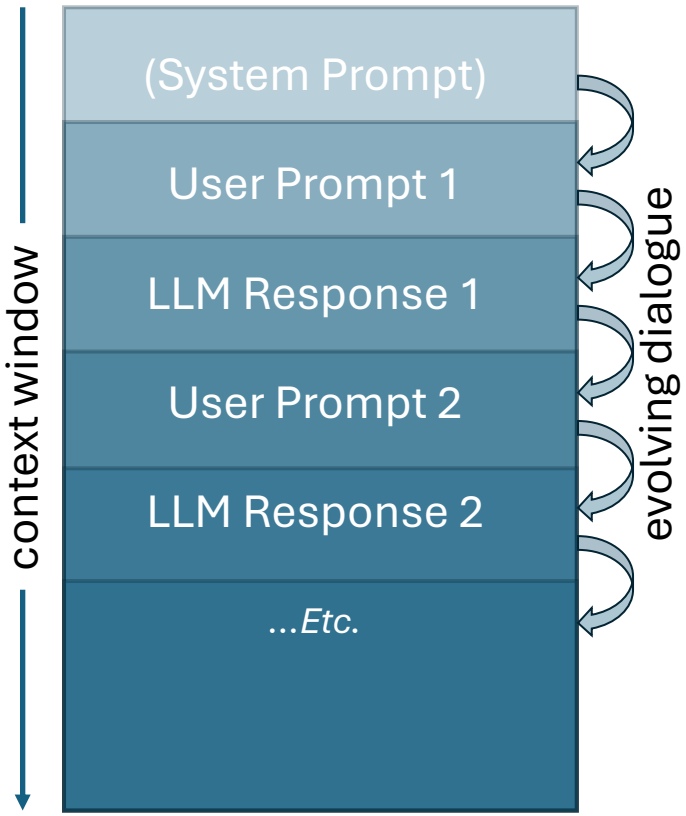
Using an LLM (1)

Autoregressive Chaining

- Repeated invocation (inference) of the LLM
- Predict the next word (token) based on the text observed so far
- One word (token) is generated at a time



Using an LLM (2)
Prompt Cycle



- Conversation is primed with system prompt (optional)
- User prompt is added to context window
- Context window is input to LLM yielding a response
- Response is added to the context window
- Context window grows for each interaction cycle

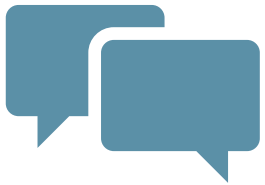
Context Window Overflow:

- Context window > context length
- LLM “forgets” earliest parts of the conversation unless action is taken

Prompt Engineering

Task

Role - The persona the AI should adopt
Command - The action verb directing the AI
Topic - Subject matter focus area
Query - Specific question to be answered



Instructions

Output - Expected attributes of generated content
Structure - Organization format, sections, flow
Do's - Acceptable qualities and content
Don'ts - Unacceptable qualities and content
Points/Ideas - Specific concepts to include
Examples - Samples to illustrate the desired output

Context

Target Audience - Intended consumer of the content
Perspective - Point of view to adopt
Purpose - Goals and motivations
Supplementary Info - Additional background details

How do LLMs work?

Concepts:

1. Tokenization
2. Embedding
3. Attention Mechanism
4. Transformers

Tokenization

- Break down complex text into manageable units (features)
- Convert text to sequence of numbers (token IDs)
- Vocabulary size (for example, 50,257 tokens in GPT-3)
- Done using a ML model separately trained from LLM
- A token is ~ ¾ of a word (GPT-3)
- *"What is the tallest building"* → `What is the tall est building`

Example Special Tokens	
BOS	Beginning of Sequence
EOS	End of Sequence
PAD	Padding Token
UNK	'Unknown' Token

Embedding

- Numerical (vector) representations of tokens
- Across many dimensions (12288 for GPT-3)
- Relationship between data in multi-dimensional space
- Each word (token) is turned into a vector capturing:
  - Semantic meaning
  - Relationship to other words
- Example:
  - '*book*' and '*worm*' are very different but for the concept of a '*bookworm*'
  - Embeddings will be close along (at least) one dimension
- Can be stored in Vector Databases

Embedding Example

Text:

‘What is the tallest building?’

Raw tokens:

[‘What', ‘is’, ’the’, ‘tall', ‘est’, ‘building’, ‘?', '[SEP]’]

Token IDs:

[101, 2009, 1005, 1055, 2058, 7706, 2692, 999]

Embeddings: tensor(

[[[0.1116, 0.0722, 0.3173, ..., -0.0635, 0.2166, 0.3236],
[-0.4159, -0.5147, 0.5690, ..., -0.2577, 0.5710, 0.4439],
[-0.4893, -0.8719, 0.7343, ..., -0.3001, 0.6078, 0.3938],
...,
[-0.2746, -0.6479, 0.2702, ..., -0.4827, 0.1755, -0.3939],
[0.0846, -0.3420, 0.0216, ..., 0.6648, 0.3375, -0.2893],
[0.6566, 0.2011, 0.0142, ..., 0.0786, -0.5767, -0.4356]]])

GPT-3:
[8 x 12288]

Attention Mechanism

Swing the bat!

vs.

The bat flew at night

John doesn't eat meat.
Pete doesn't like steak.
Who's vegetarian?

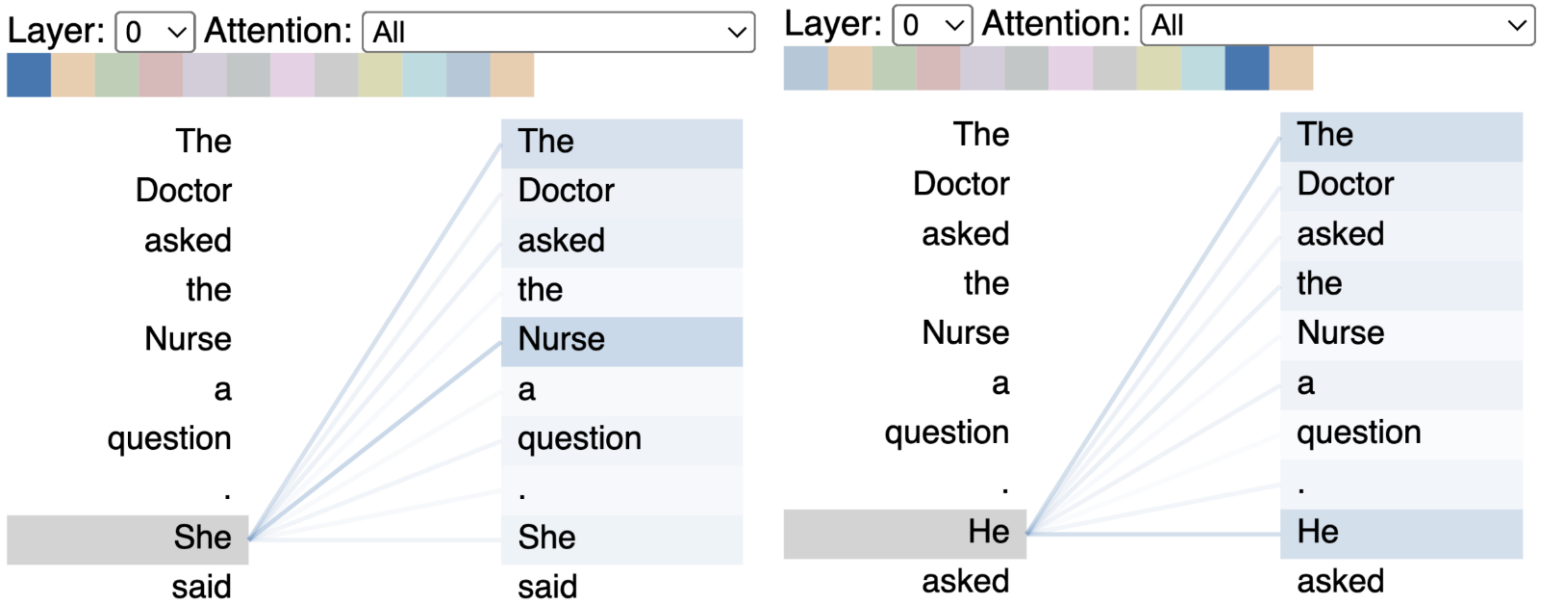


Image: Explainable AI – Visualizing Attention in Transformers

Compute context dependent weights from embeddings

Transformers

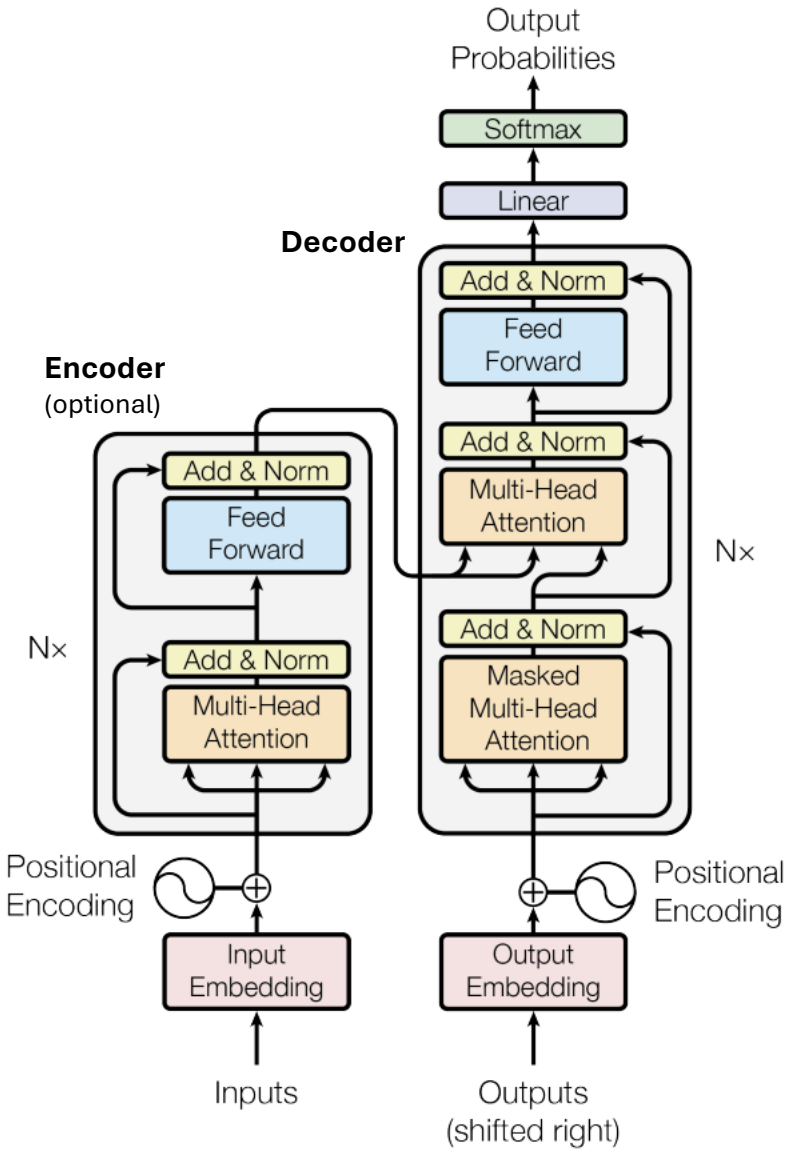
tokens →

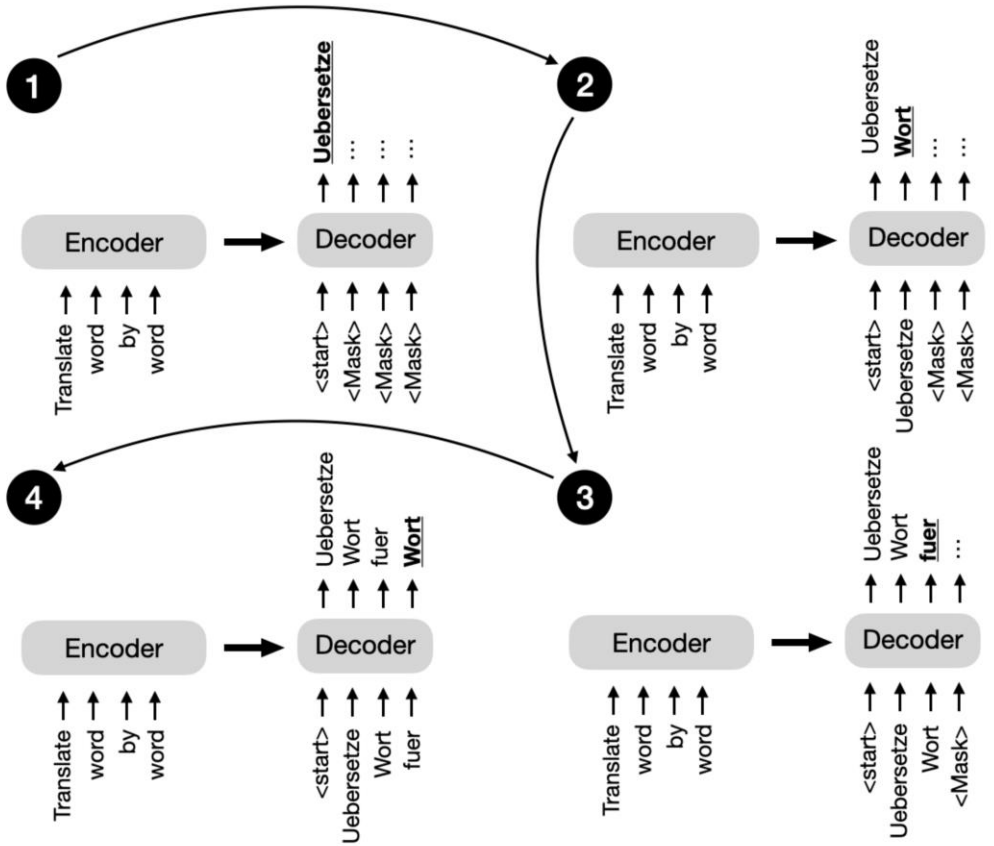
multi-layer perceptron →

attention →

embedding →

tokens →





Original transformer architecture from "Attention is all you need", Google DeepMind, 2017

LLM Training

Semi-Supervised Learning

Stage 1: Pre-Training

- No data labeling needed (next word is the label)
- High quantity of generic (internet) data
- Base model with generic language abilities

Stage 2: Fine-tuning

- Data labeling by human beings
- Need quality data, typically QnA
- Become an “Assistant” in a certain field

Stage 3 (optional): RLHF

- Reinforcement Learning from Human Feedback
- Use model to generate different responses
- Human being selects the best one for fine-tuning

Examples:

- Text classification
- QnA
- Summarization
- Text generation
- Coding

LLMs – Limitations & Challenges

HOT CHIPS

- Struggle in Math, Logic, Reasoning
- Bias & Safety
- Model censorship
- Knowledge only up until time of training - party addressed using tools + RAG
- Hallucinations - LLMs can be confidently and convincingly wrong
- Cost – Power consumption for training and inference
- Legal & ethics – Ok to train on copyrighted material? Fair use?
- Malintent Use – Misinformation & disinformation campaigns, fakes, scams, etc.
- Workforce disruptions – example: Hollywood labor genAI concerns
- Future AGI? – Artificial General Intelligence: what if AI starts thinking for itself?

LLMs Future, Examples

HOT CHIPS

- Fact check themselves using information from Web
- Mixture of Experts – Models built using models
- Multi modality – voice, images, video input and output
- Improve reasoning – “think slowly”
- Larger context windows (process longer prompts)
- Extend LLMs with external memory

PrimisAI

HOT CHIPS

2. Introduction to RAG

What is RAG?

- ***R****etrieval* ***A****ugmented* ***G****eneration*
- Meta - ArXiv Patrick Lewis et. al., May 2020 (https://doi.org/10.48550/arXiv.2005.11401)
- Enable LLM to lookup information
- Leverage data the LLM may not have been trained on
- LLM acts on input prompt in combination with the RAG data

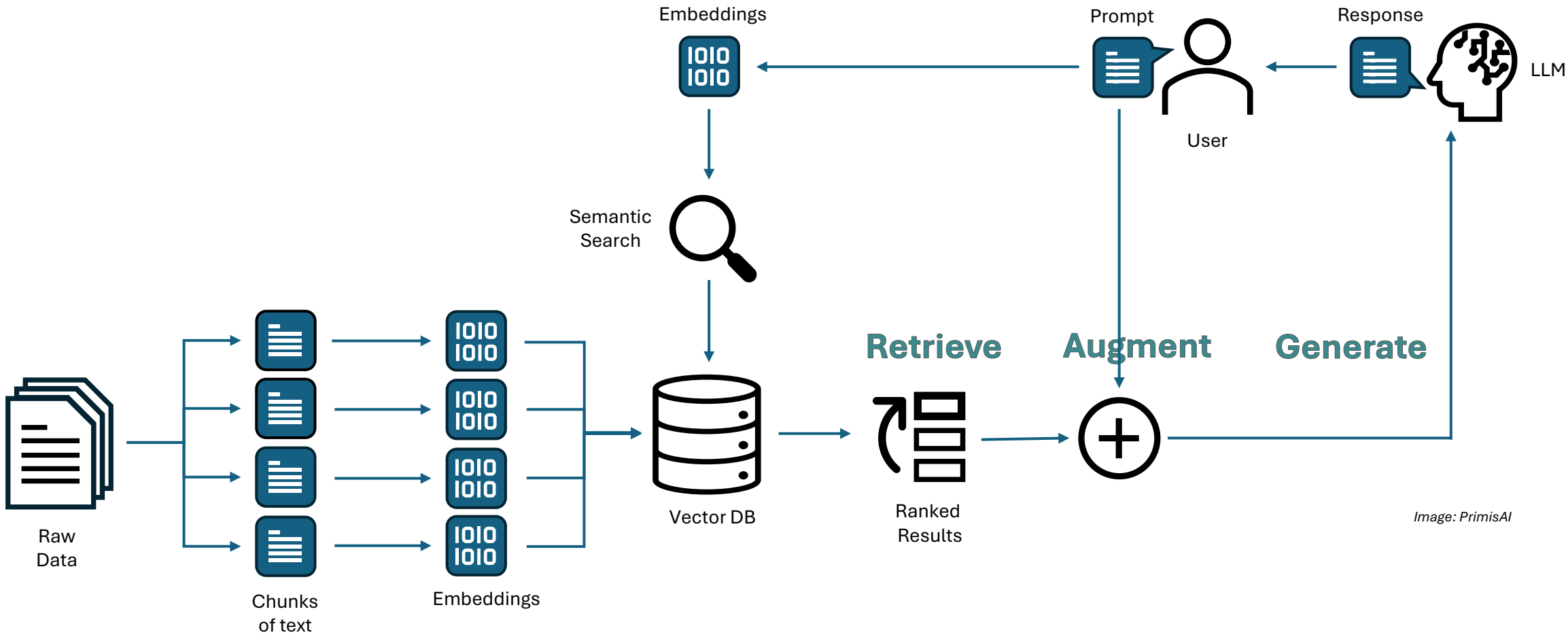
Why RAG?

>95% of the world's data is private!

- Ability to empower the LLM (which is frozen in time!) with:
  - Proprietary information – for example semiconductor IP
  - Recent data not known when the LLM was trained
  - Known to be relevant facts → reduce hallucinations

- Possibly equip the LLM with long-term memory
  - Break-through context window limitation

RAG Architecture

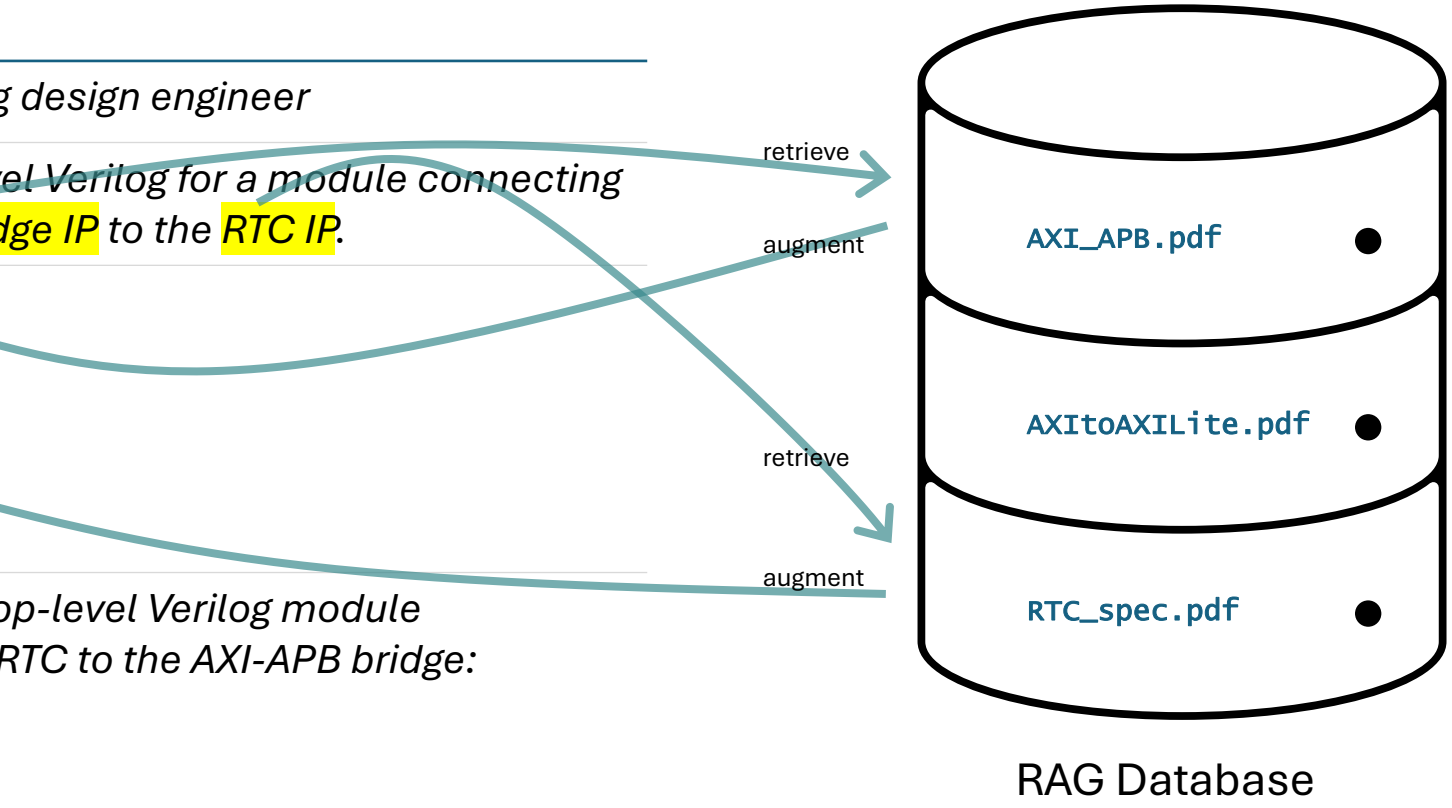


RAG Example

Instantiate IP from IP-Catalog

context window

System prompt:	*You are a Verilog design engineer*
User prompt:	*Propose top-level Verilog for a module connecting the AXI-APB bridge IP to the RTC IP.*
Augmented from RAG:	module AXI_APB(input clk, ...); module RTC(input clk, ...);
Response:	*Sure, here is a top-level Verilog module connecting the RTC to the AXI-APB bridge:* module APB_RTC_top(input clk, ...);



3. LLM+RAG → EDA AI-Agent

LLM-Based AI Agent

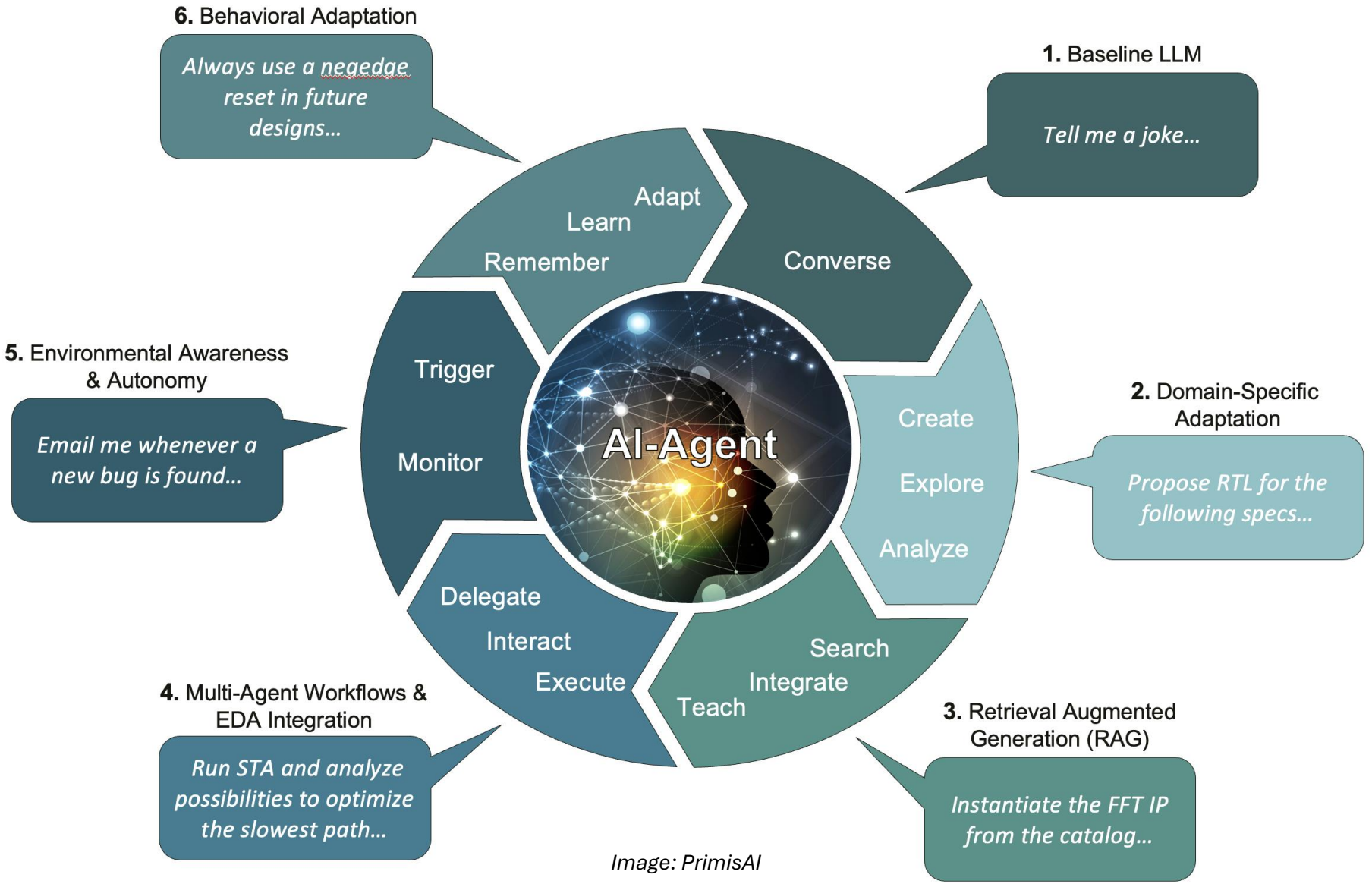


Image: PrimisAI

Multi Agent Workflow Automation

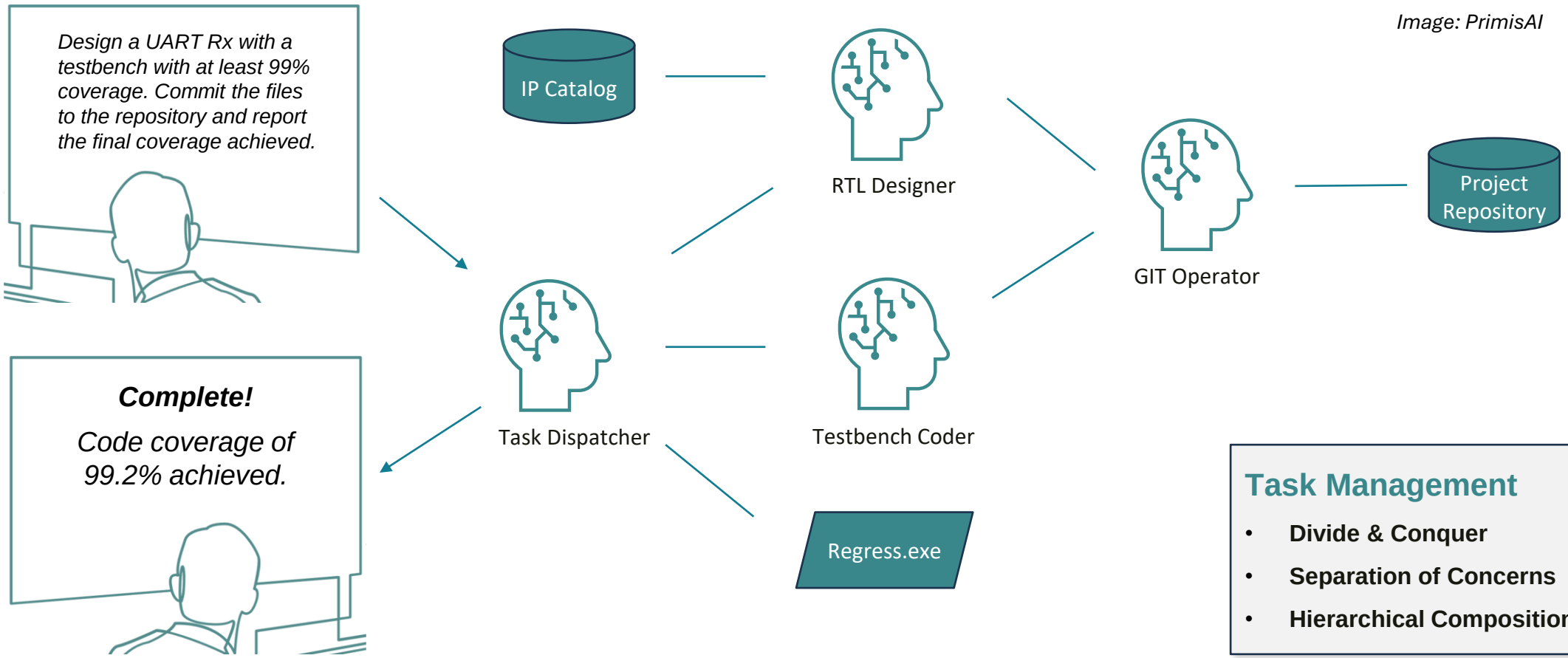


Image: PrimisAI

Task Management

- **Divide & Conquer**
- **Separation of Concerns**
- **Hierarchical Composition**

LLMs EDA Promise

Reducing the Human-in-the-Loop need

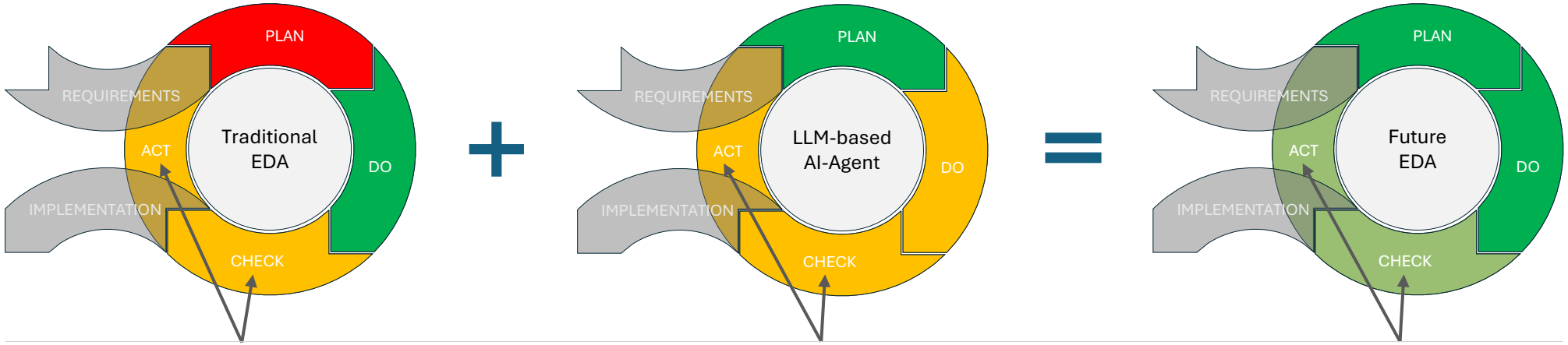


Image: PrimisAI

PrimisAI

HOT CHIPS

Thank You!