2025
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 24-27, 2025

Towards Automated Verification IP Instantiation via LLMs

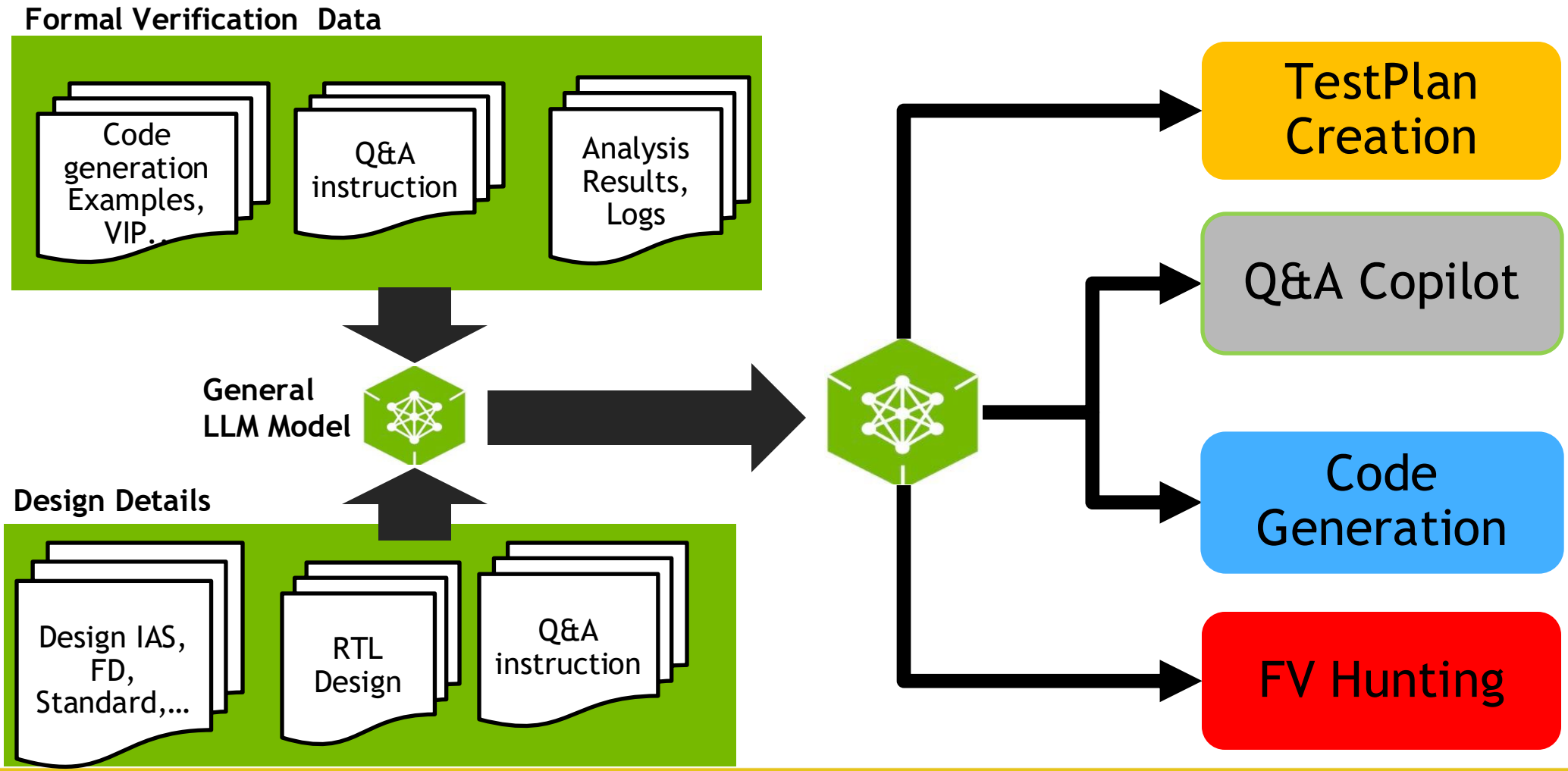
Ghaith Bany Hamad, Michael Marcotte, Syed Suhaib

Nvidia

NVIDIA

accellera SYSTEMS INITIATIVE

How LLMs will improve FV workflow?

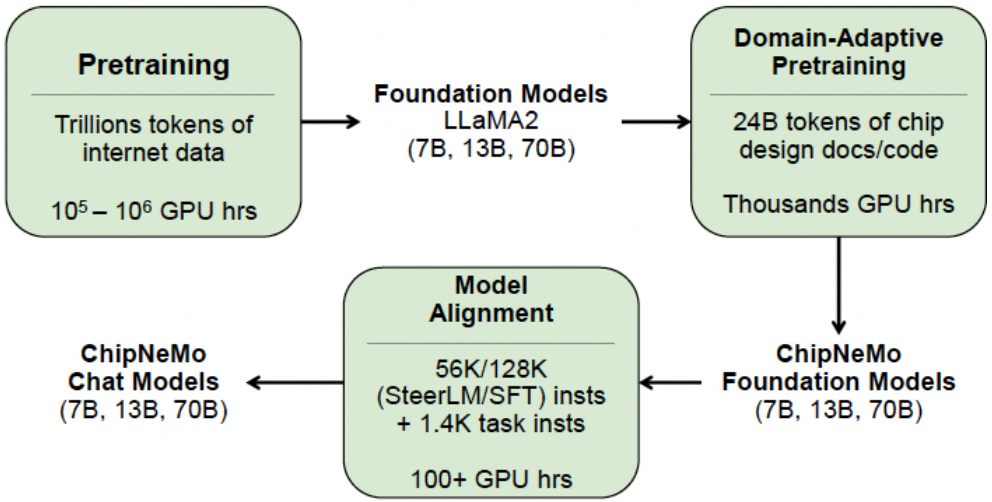


Benchmarking LLM Capabilities

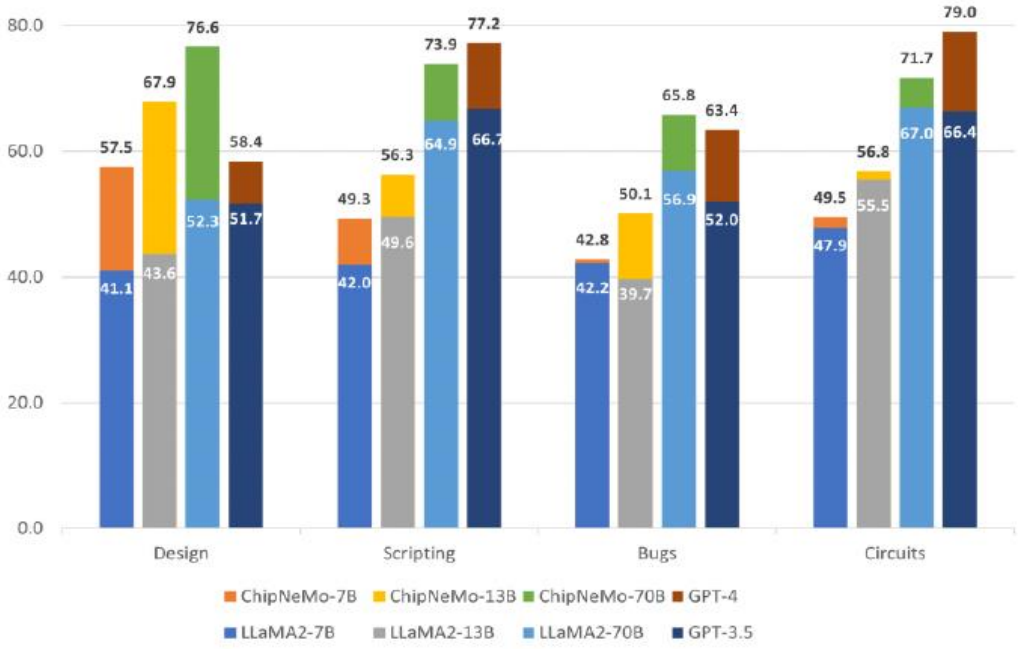
- **Performance Evaluation**:
  - Assessing how well the LLM performs on specific tasks, such as NL understanding, generation of SVAs, code analysis and summarization, and more.
- **Comparative Analysis**:
  - Comparing different LLMs or different versions of the same model to identify which one performs better under certain conditions or tasks.
- **Accuracy and Correctness**:
  - Generates syntactically correct and semantically meaningful code.
- **Code Quality**:
  - Measuring code readability, maintainability, and efficiency.
- **Task Completion**:
  - Writing assertions, setup config, and full FV testbench, given specific requirements or prompts.

Related Case Study: Domain-Adapted LLMs for VLSI Design and Verification

The training flow for ChipNeMo DAPT and model alignment, including SFT



ChipNeMo evaluation results on EDA-specific benchmark, compared against other LLMs.



Domain-specific pre-training dataset (DAPT):

- Collection of proprietary hardware-related code (RTL, verification testbenches, etc)
- Natural Language (NL) datasets: hardware specifications, documentation, etc

FVEval – LLM Benchmark for FV Tasks

Datasets:

- Three benchmarking tasks:
  - NL2SVA-Human
  - NL2SVA-Machine
  - Design2SVA

Evaluation Flow

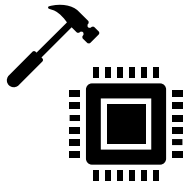
- Integrates FV tools for end-to-end automatic evaluation
- Holistic evaluation of LLM's generated assertion using property equivalence checking

Evaluating LLM's Capabilities for FV code

Flow Diagram of the FVEval Workflow

Dataset Generation

Collection of Human-Generated Data



Synthetic Data Generator

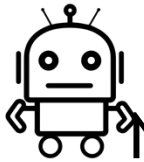
Data Preprocessing
Package input data for each subtask

LLM Inference Launcher

```
class BenchmarkLauncher(object)
```

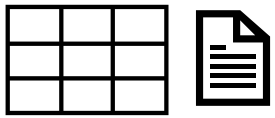
Configure Experiment

```
generate_system_prompt()
generate_user_prompt()
get_cot_strategy()
```



LLM Inference w/ ICL & CoT

```
parse_code_response(lm_response_str: str)
package_testbench(row: InputData, lm_response: str)
```

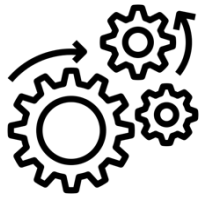


Process & Save Response

```
save_results(self,
model_name: str, results: list[LMResult])
```

Auto-Evaluation

```
class Evaluator(object)
```



```
set_tcl_file_path()
write_design_sv()
write_testbench_sv()
```

JasperGold -batch

HF Evaluate

```
evaluate_jg
(result_list:
list[LMResult],
with_rtl_design: bool)
```

```
evaluate_text_similiar
ty(result_list:
list[LMResult])
```

Process & Save Results
Parse JG output string
Save as .csv file

NL2SVA-Machine (Direct Low-Level NL -> SVA Assertion)

Metrics:

- Syntax correctness
- Functional correctness:
  - **BLEU** score: n-gram similarity between ground-truth solution vs. LM solution
    - Proxy Measure for functional similarity

Model Name	0-shot		2-shot	
	Syntax Pass@1	BLEU	Syntax Pass@1	BLEU
Mixtral-8x7B	0.152	0.189	0.747	0.283
LLaMA2-70B	0.245	0.283	0.808	0.283
gpt-3.5-turbo	0.320	0.359	0.919	0.444
gpt-4	0.521	0.875	0.960	0.465
ChipNemo70B (DAPT+DSFT)	0.425	**0.864**	0.980	**0.495**

NL2SVA-Human results (Testbench + High-Level NL to SVA Assertions)

Metrics:

- Syntax correctness
- Functional correctness:
  - Signal Match
  - BLEU score

Model Name	0-shot			3-shot		
	Syntax Pass@1	Signal Match	BLEU	Syntax Pass@1	Signal Match	BLEU
Mixtral-8x7B	0.626	0.0	0.286	0.808	0.162	0.421
LLaMA2-70B	0.556	0.0	0.331	0.707	0.091	0.457
gpt-3.5-turbo	0.354	0.0	0.254	0.889	0.313	**0.524**
gpt-4	**0.980**	0.0	0.329	0.990	0.323	0.526
ChipNemo70B (DAPT+DSFT)	0.828	**0.0**	**0.348**	**1.0**	**0.333**	0.516

Key Takeaways

- Lack of Domain-Specific Context Awareness
- Limited Understanding of Temporal Logic
- Overlooking Corner Cases
- Quality Gap: LLM vs. Human Assertions

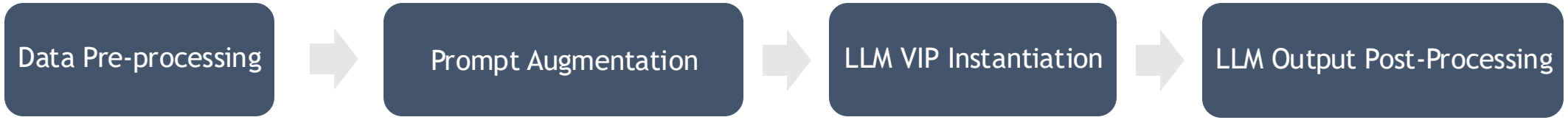
Given the limitations of LLMs, where should formal verification teams prioritize deploying LLMs to augment their workflows effectively?

LLM based VIP Instantiation Challenges

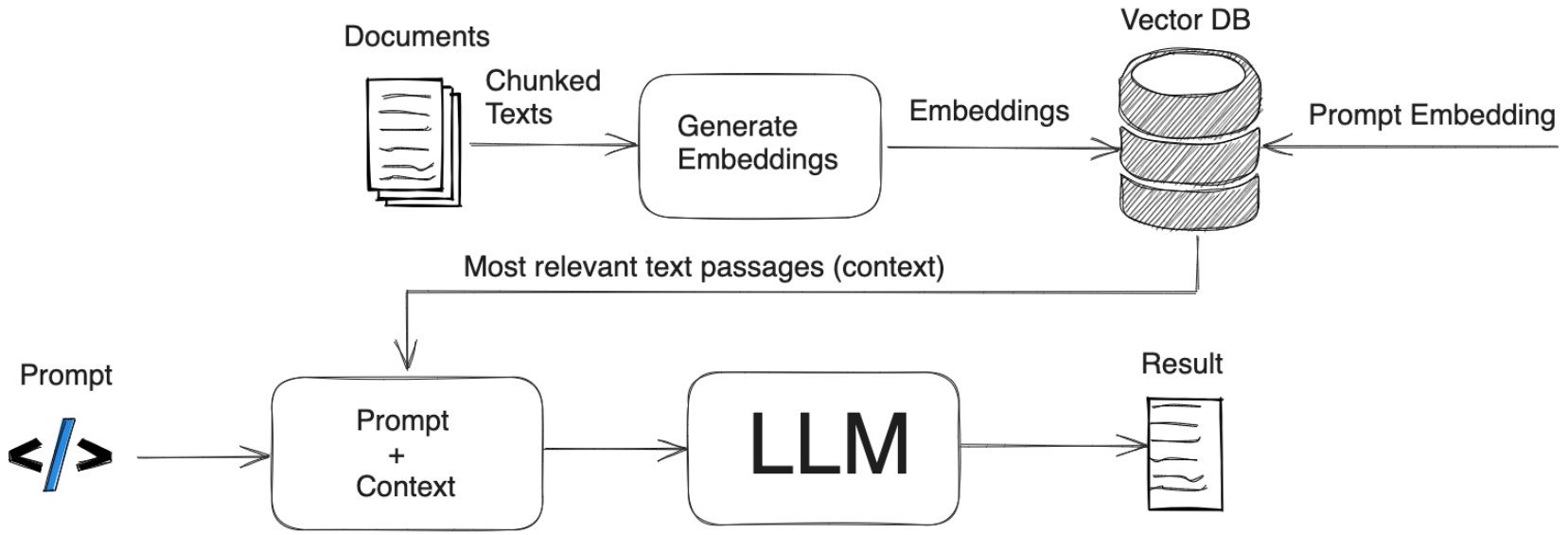
- LLM cannot read the entire file at once, so it will never get the full context of the code
  - Divide full code file into smaller partitions to iteratively feed to LLM so it effectively instantiates VIPs for entire files
  - Developed algorithm to chunk code cleverly to preserve code syntax and structure in chunking
- General models lack knowledge of the target internal VIPs
- Module parameters are buried in different files, resulting in unknown signals
  - Automatic data retrieval process to fix LLM responses in post-processing

LLM based VIP Instantiation Challenges

Approach: Design a Scalable, Modularized, Fully Automated Flow

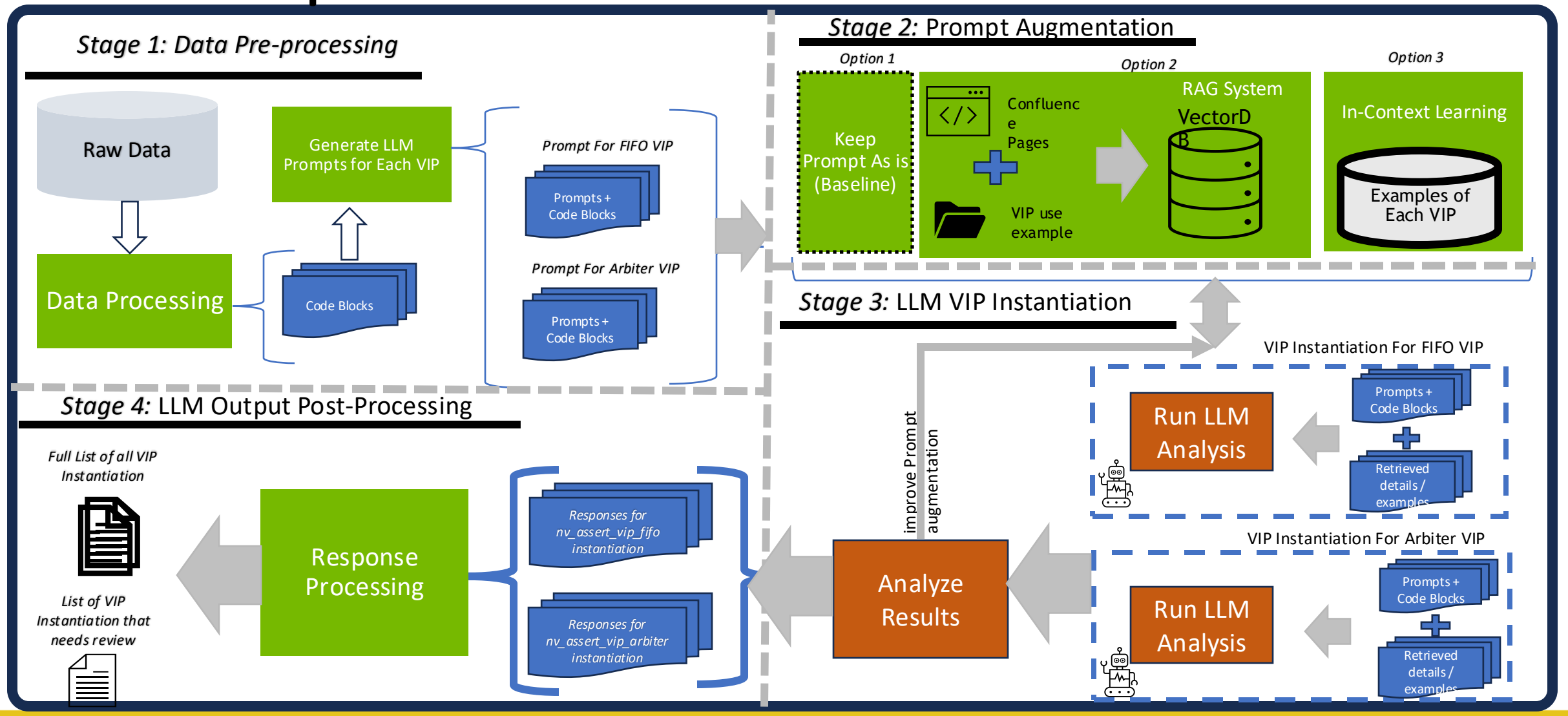


RAG Based Chatbots



- Avoids having to Train/Fine-Tune LLM
- Avoids Hallucinations – As keeps the LLM grounded to Truth in the presented Context

Proposed Flow VIP Instantiation



VIP Instantiation Flow Results

FV VIP	LLM Model	Prompt Augmentation	Correct	Incorrect	Missing	Total # of instances
nv_assert_vip_fifo	chipmixtral_8x7b_chat_tp4_trt_h100	Baseline	0	0	7	7
	chipmixtral_8x7b_chat_tp4_trt_h100	ICL	4	1	2	7
	Llama-3-70b	RAG	7	0	0	7
nv_assert_vip_arb	chipmixtral_8x7b_chat_tp4_trt_h100	Baseline	0	0	12	12
	chipmixtral_8x7b_chat_tp4_trt_h100	ICL	1	4	7	12
	Llama-3-70b	RAG	11	1	0	12

Table 1: Results of the VIP instantiation for both nv_assert_vip_fifo and nv_assert_vip_arb.

Key for Table 1:

Correct:	This column means completely correct instantiation, all signals correct
Incorrect:	This column means VIP instantiation for modules present with either incomplete or wrong signals
Missing:	This column means missing VIP instantiation for module

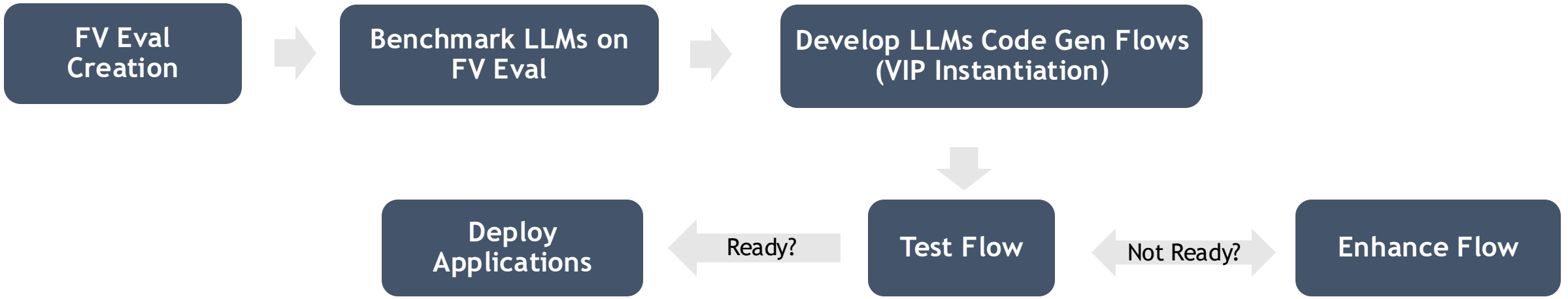
Key Takeaways

- **Superior Performance of RAG**: Retrieval-Augmented Generation (RAG) achieved the highest accuracy in VIP instantiation, outperforming Baseline and In-Context Learning (ICL).
- **100% Accuracy for FIFO VIPs**: RAG correctly instantiated all FIFO VIPs (7/7 correct instances).
- **High Accuracy for Arbiter VIPs**: RAG successfully instantiated 11 out of 12 arbiter VIPs, with only one incorrect result.
- **Baseline Model Ineffectiveness**: The baseline approach (without augmentation) failed to instantiate any VIPs correctly.
- **ICL Shows Moderate Improvement**: ICL improved performance over the baseline but was inconsistent, especially for arbiter VIPs (1 correct, 4 incorrect, 7 missing).
- **Llama-3-70b Outperforms Finetuned Model**: The Llama-3-70b model with RAG significantly outperformed the fine-tuned chipmixtral_8x7b_chat_tp4_trt_h100 model.
- **Reduced Hallucinations**: The RAG approach minimized incorrect instantiations compared to other methods.

Conclusion

- **Automated VIP Instantiation**: LLMs streamline and automate the integration of Verification IPs (VIPs), reducing manual effort.
- **Retrieval-Augmented Generation (RAG)**: Achieves superior accuracy in VIP instantiation compared to baseline and In-Context Learning (ICL).
- **Efficiency Gains**: Significant reduction in verification time by automating common VIP deployments.
- **Scalability**: Modular framework adaptable for various VIPs without additional model fine-tuning.
- **Improved Verification Coverage**: More reliable VIP instantiations reduce missed bugs and enhance verification integrity.
- **Optimized Prompt Engineering**: Data pre-processing pipeline ensures efficient LLM usage and minimal hallucinations.
- **Performance Validation**: RAG-based approach outperformed other methods in accuracy and reliability.

Next Steps



How to Enhance Existing Flow?

- Add more Data to the RAG VD
- Upgrade driving LLM (currently Llama3-70b or DeepSeek)
- Additional support for more VIPs (forward progress VIPs)
- Deploy to VIP instantiation flow to more projects

Questions