Large Language Model-Driven Real-World Applications: An Automated IC Testing System

Weimin Fu*, Gang Qu†, Xiaolong Guo*
*Department of Electrical and Computer Engineering, Kansas State University
†Department of Electrical and Computer Engineering, University of Maryland
weiminf@ksu.edu, gangqu@umd.edu,guoxiaolong@ksu.edu

Abstract

Large language models (LLMs) have rapidly evolved from simple text-based interactions to accepting and generating multi-modal information, including images and sound. This advancement opens up new avenues for application in real-world interactions. We have integrated large language models with automated testing systems to explore these possibilities. Our preliminary experiments, which focused on testing Analog-to-Digital Converters, demonstrate the applications' feasibility and potential scope. Through these experiments, we aim to highlight how LLMs can unlock new hardware design and verification process capabilities.

Index Terms

Hardware Auto Testing, Large Language Model, Generative AI

I. Introduction

Artificial intelligence assistants based on large language models have become the focus of academia, industry, and everyone. With the transformation of large language models (LLMs) into large multimodal models (LMMs), the hot topic has become how to stimulate more capabilities and applications by providing the models with more input types. Automated testing is an excellent entry point for hardware design verification processes and endows LLM with the ability to interact with reality. Also, employing generative AI technologies has the potential to lower labor costs and enhance the efficiency of hardware testing processes. We have demonstrated the effectiveness of LLMs in the domain of hardware security, as evidenced by our previous works [1], [2]. Building on this foundation, in this demo, we constructed an automated integrated circuits (ICs) testing platform to demonstrate the potential applications of this LLM-based approach across the field. This platform takes the definitions of inputs and outputs from the IC's datasheet and the eval board as inputs, uses a meticulously designed experimental procedure, leverages decisions made by the PC and LLM to control the behavior of the instrumentation, and automatically provides evaluation results.

II. Methodology and Experiment

As illustrated in Fig.1, the system architecture encompasses four core components: the LLM service, local automated testing scripts, instrumentation, and the Device Under Test (DUT).

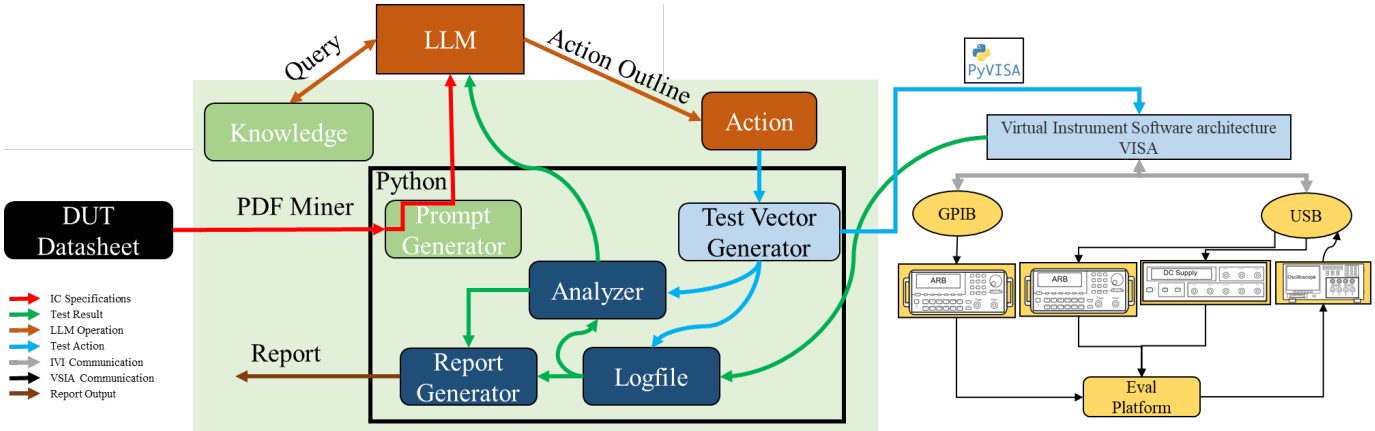

Fig. 1. LLM-Enhanced Automated Testing Platform Architecture: On the left, the DUT Datasheet includes the datasheet for the IC under test and the interface definitions of the evaluation board; the orange LLM at the top represents ChatGPT used in this experiment; the light green box illustrates the tools we developed: 1. Knowledge and experience repository for hardware testing, 2. Prompt templates for hardware testing, 3. LLM Action Outline to ensure the correctness of LLM output formats, 4. Generation of test vectors and their transmission to instrumentation, 5. Analysis and logging; the bottom part shows the connection to various instrument devices through VISA, which in turn connect to the evaluation board.

The initial step involves deconstructing the datasheet, including the datasheet for the DUT IC and the evaluation board's datasheet. We employ PDFMiner to extract crucial textual and tabular information from the datasheets. The prompt-extracted information is fed into OpenAI's GPT-4 API, allowing the LLM to access local DUT data and information about the testing platform. This process ensures that the LLM comprehends the entity's control and interaction. We meticulously designed these prompts to accurately capture and interpret information extracted from the datasheets, including integrated circuit interface definitions, input/output ranges, and performance benchmarks. We reuse the details in the automated testing scripts as strict boundaries to prevent damage to the DUT from incorrect data ranges.

Following this, we construct action interfaces based on the OpenAPI specification, strictly limiting the format of GPT-4's outputs to ensure the generated test vectors meet technical standards and offer high reliability and repeatability.

Leveraging the PyVISA and IVI libraries, we control various test instruments such as signal generators, oscilloscopes, and power supplies according to the test plan, setting testing conditions and acquiring results. Fig. 2 depicts the output from the instrumentation during operation, which is also uploaded to the PC. Compared to expected datasheet results, preliminary local script analyses of the test outcomes are uploaded to a cloud-based large language model for in-depth analysis. The cloud model evaluates the correctness of the test results and provides recommendations for subsequent tests, forming closed-loop feedback until all testing metrics meet the termination criteria.

Upon completing all tests, the system automatically generates a detailed report, including results, comparative analysis, and potential improvement suggestions. All test data and logs are meticulously preserved for future in-depth analysis and review.

As shown in Table I, most of our instrumentation is connected to the PC via USB, except the 33250A used for generating analog signals, which utilizes GPIO due to the absence of a USB interface. Despite these differences in connection methods, the underlying protocol for all devices remains consistent and is based on VISA.

TABLE I
HARDWARE CONFIGURATION

Category	Device Name	Model	Purpose	PC Connection Type
Power Supply	Keithley Triple Channel DC Power Supply	2230G-30-1	Supplies VDD1 and VDD2 power	USB
Oscilloscope	Tektronix Mixed Domain Oscilloscope	MDO3014	Observes waveforms at VDD1 test point and MDAT port output	USB
Signal Generator	Agilent Function/Arbitrary Waveform Generator	33220A	Provides MCLKIN clock signal	USB
Signal Generator	Agilent Function/Arbitrary Waveform Generator	33250A	Provides analog signal input (VIN+)	GPIO
ADC	ADuM7703 16-Bit, Isolated, Sigma-Delta ADC	ADuM7703	IC under test	-
Evaluation Board	ADuM7703 Evaluation Board	-	Testing platform	-

Furthermore, as outlined in Table II, our system also relies on a suite of software and libraries for data extraction, information interpretation, and automated control. This includes PDFMiner for extracting information from datasheets, the OpenAI API for interactions with large language models, and both PyVISA and NI-visa for the control of hardware devices.

TABLE II
SOFTWARE CONFIGURATION

Category	Software Name	Version	Purpose
Data Extraction Tool	PDFMiner	20191110	Extracts text, images, and tables from IC datasheets
Large Language Model API	OpenAI	1.12.0	Extracts IC interface definitions, input/output ranges, and calibration performance
API Specification	OpenAPI Specification	3.1.0	Defines the interface for interaction with the large language model, ensuring compatibility with backend automation scripts
Automated Testing Control	PyVISA	1.14.1	Controls testing equipment such as signal generators and oscilloscopes
Driver Interface	ivi	1.14.1	Provides a standard interface for communication with instruments
Driver Program	NI-visa	25165824	Provides underlying hardware control support for PyVISA

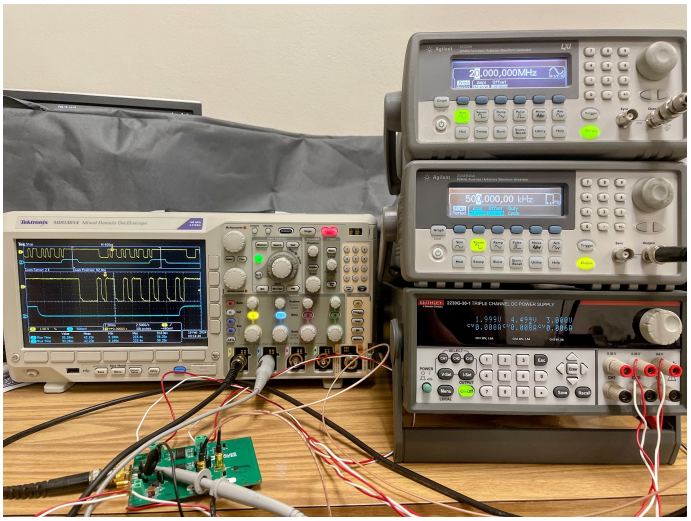

Fig. 2. The display of the test tool during automated testing. In the figure, the connections are still manually made. Power sources VDD1 and VDD2 are connected to the Keithley Triple Channel DC Power Supply, MCLKIN is connected to the Agilent Function/Arbitrary Waveform Generator 33220A, VIN+ is connected to the Agilent Function/Arbitrary Waveform Generator 33250A, Oscilloscope channel 1 is connected to MDAT, and channel 2 is connected to the VIN+ test point on the eval board. At this time, on the test board, Link LK1 is set to Position B, LK2 is in Position C, LK3 is in Position C, LK4 is set to Position B, LK5 is removed, LK7 is inserted, and LK8 is removed. The automated script operates the instruments.

REFERENCES

[1] W. Fu, S. Li, Y. Zhao, H. Ma, R. Dutta, X. Zhang, K. Yang, Y. Jin, and X. Guo, "Hardware phi-1.5 b: A large language model encodes hardware domain specific knowledge," 29th Asia and South Pacific Design Automation Conference(ASP-DAC), 2024.

[2] W. Fu, K. Yang, R. G. Dutta, X. Guo, and G. Qu, "Llm4sechw: Leavering domain-specific large language model for hardware debugging," Asian Hardware Oriented Security and Trust (AsianHOST), 2023.