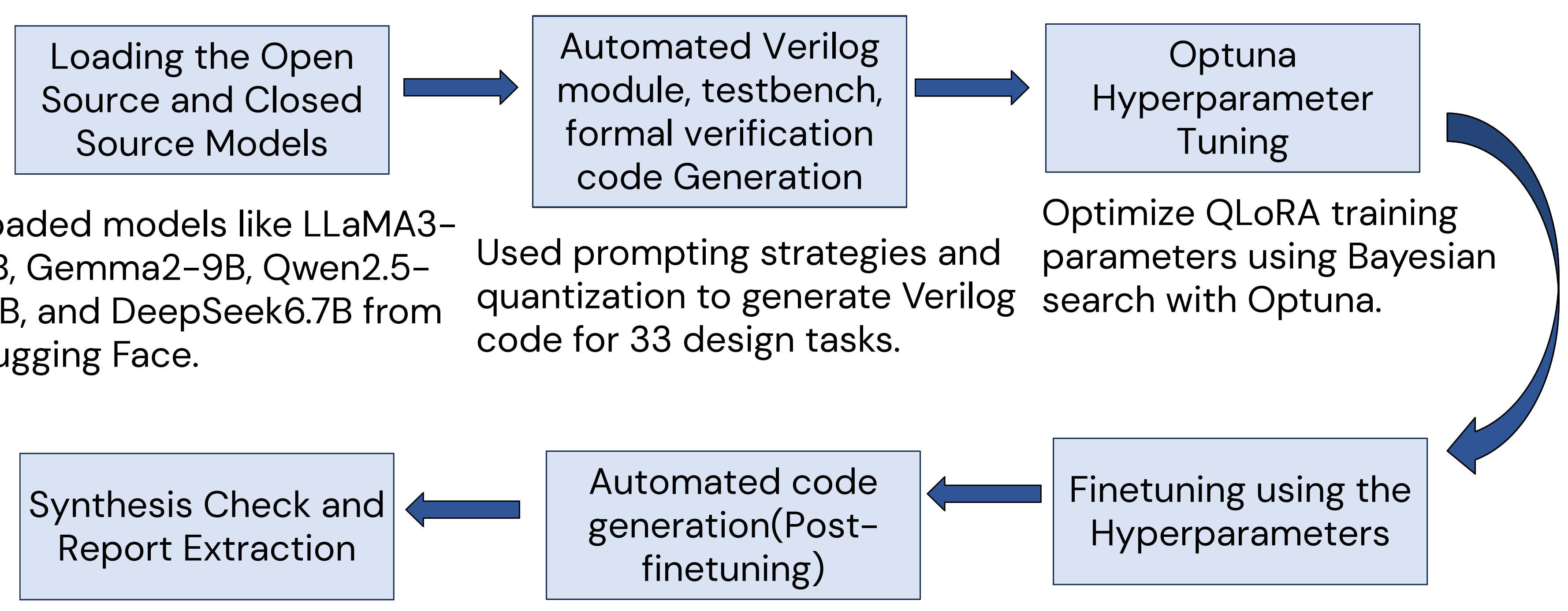
Benchmarking Open-Source Large Language Models for Verilog Code Generation and Synthesis

Sia Jariwala,Ginisha Garg
Under the guidance of Prof. Joycee Mekie, Mihir Agarwal, Zaqi Momin
Department of Electrical Engineering, Indian Institute of Technology Gandhinagar

Motivation

- Writing Verilog HDL modules, testbenches, and assertions is time-consuming, error-prone, and requires domain expertise.
- While closed-source LLMs like DeepSeek show strong capabilities in code generation, they lack transparency, offer limited flexibility, and often involve high costs.
- Open-source large language models (LLMs) provide a promising alternative for Electronic Design Automation (EDA) by enabling:
1. Fine-tuning approaches focused on HDL design tasks
2. Scalable automation using efficient, quantized models
- This project investigates whether fine-tuned open-source models can match or outperform closed-source baselines in Verilog code generation and synthesis — enabling more accessible and adaptable AI tools for hardware design workflows.

Methodology


Loading the Open Source and Closed Source Models
Automated Verilog module, testbench, formal verification code Generation
Optuna Hyperparameter Tuning
Finetuning using the Hyperparameters
Automated code generation(Post-finetuning)
Synthesis Check and Report Extraction

Loaded models like LLaMA3-8B, Gemma2-9B, Qwen2.5-14B, and DeepSeek6.7B from Hugging Face.

Used prompting strategies and quantization to generate Verilog code for 33 design tasks.

Optimize QLoRA training parameters using Bayesian search with Optuna.

Fine-tune models on the MG-Verilog dataset with the best Optuna config.

Regenerate Verilog code with the fine-tuned models for evaluation.

Run Vivado synthesis in batch mode and extract timing, power, and LUT reports.

TABLE I: Verilog Designs Used for Benchmarking

Half Adder	Full Adder	Parameterized Adder	Parameterized Subtractor	Parameterized Multiplier	Parameterized Divider	32-bit Register File	Parameterized Shift Register	4-bit ALU	RAM	ROM
FSM (1100 pattern)	Fibonacci Number	Binary Adder Tree	Ternary Adder Tree	Dual Clock Synchronous RAM	Single Clock Synchronous RAM	RAM with Separate Input and Output Ports	Single-Port RAM	Counter with Asynchronous Reset	Bidirectional Pin	UART
FFT Module	Digital Filter	BCD to Gray converter	7 Segment LCD	Gray Counter (Design 1)	Behavioral Counter (Design 2)	Parameterized Comparator (Design 3)	Modulation and Demodulation (Design 4)	Parameterized Counter (Design 5)	True Dual-Port RAM with a Single Clock (Design 6)	8x64 Shift Register with Taps (Design 7)

Prompting Techniques

Zero-shot prompting
- **No examples, just the task.**
- **Relies wholly on the model's pretraining without prior examples.**

Few-shot Prompting
- **Prompt includes input–output Verilog examples.**
- **We used Half Adder and Parameterized Adder as guiding examples.**

Chain-of-Thought (CoT) Prompting
- **Model reasons step-by-step.**
- **Boosts reasoning but may cause the model to overthink simple tasks.**

Analysis

Synthesis Success Rate
- Gemma2-9B (fine-tuned) and Qwen2.5-14B (int8) both achieved 81.82% success (27/33).
- Fine-tuned open-source models outperformed or matched DeepSeek in several configurations.

Power Usage
- LLaMA3-8B (int8 + few-shot) used just 1.98W, synthesizing 26 designs.
- DeepSeek (int4 + CoT) had lowest power (0.28W) but only 6 successful designs.

Datapath Delay
- Gemma2-9B (int4 + CoT) had the lowest delay at 4.41 ns.
- DeepSeek models often exceeded 7.7 ns delay, showing inefficiency.

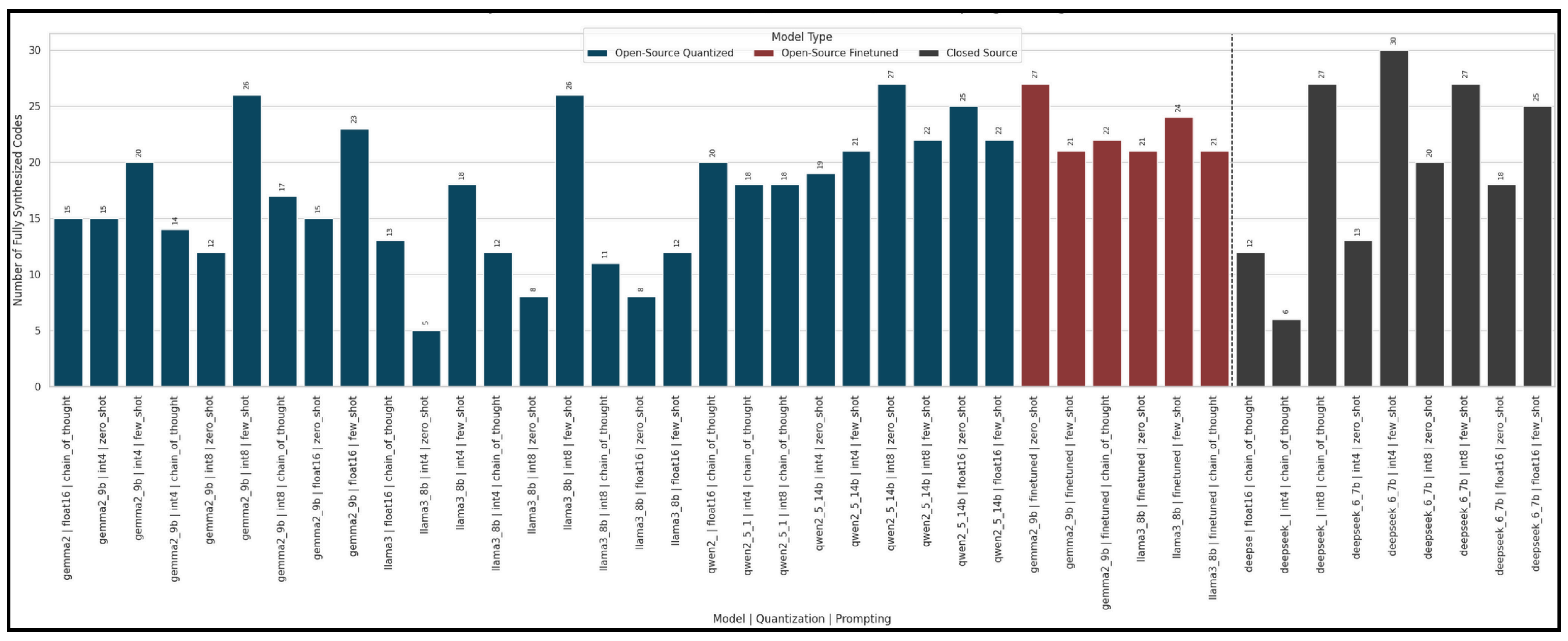
LUT Usage
- While many LLaMA3-8B and Gemma2-9B configurations maintained LUT usage below 40, there were some outliers, such as LLaMA3-8B (float16 + zero-shot).
- DeepSeek (float16 + CoT)—produced verbose designs, consuming over 80 LUTs.

Conclusion

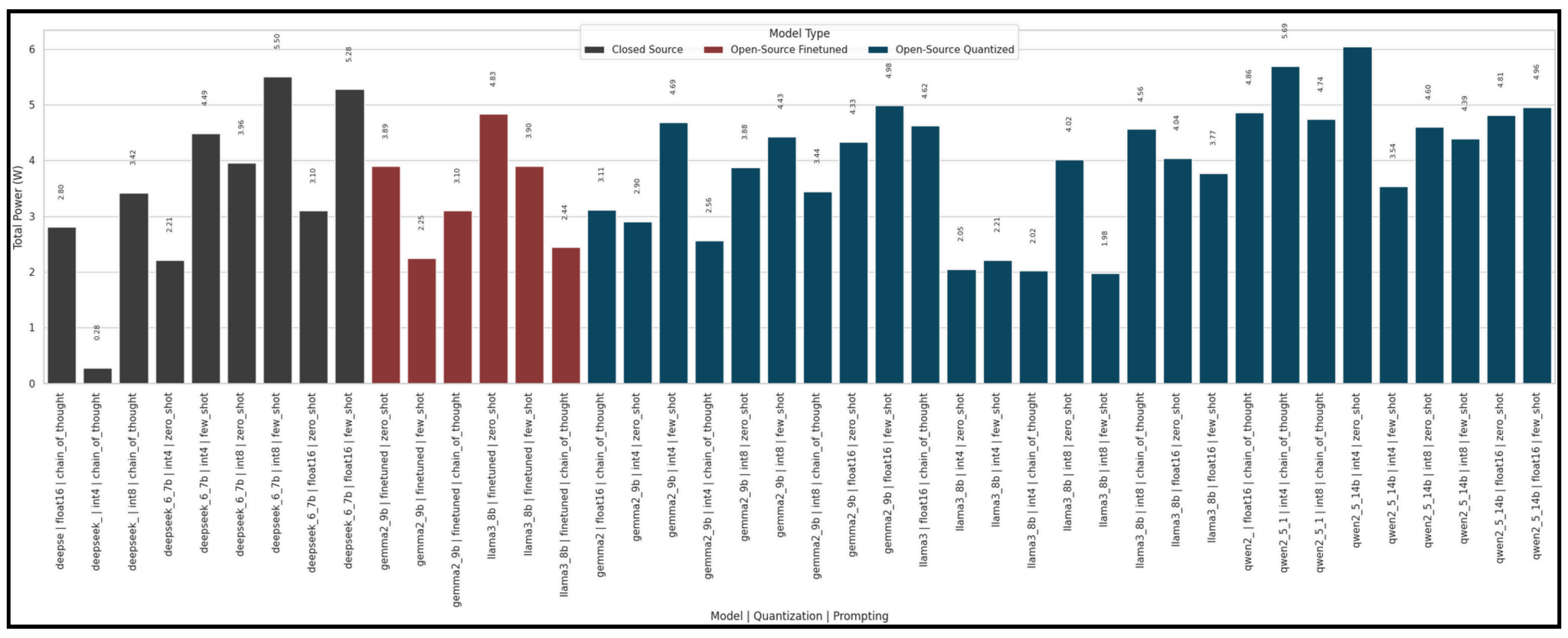
- Open-source LLMs, when fine-tuned and paired with effective prompting strategies, can match or even surpass closed-source models like DeepSeek in Verilog code generation and synthesis.
- Models such as Gemma2-9B and LLaMA3-8B achieved high synthesis success rates (up to 81.82%) with lower power usage and faster datapath delays.
- Qwen2.5-14B, even without fine-tuning, achieved 81.82% success using zero-shot prompting with int8 quantization, highlighting its strong pretrained capabilities.
- These models performed well even under quantized settings (int4/int8), making them suitable for GPU-limited and hardware-constrained environments.
- The fully automated pipeline—from prompting to synthesis—enables scalable, efficient, and reliable HDL development using open models.

Results

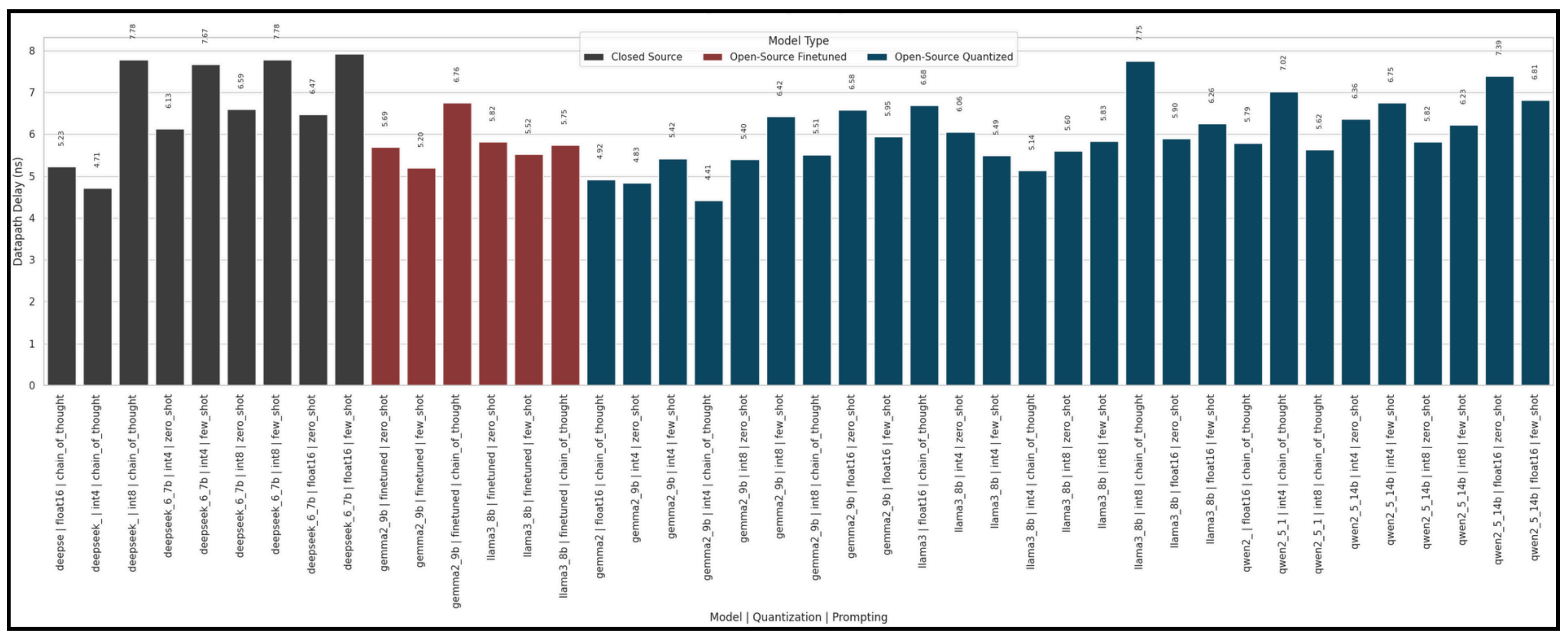
- Comparison of Synthesis Success Rate



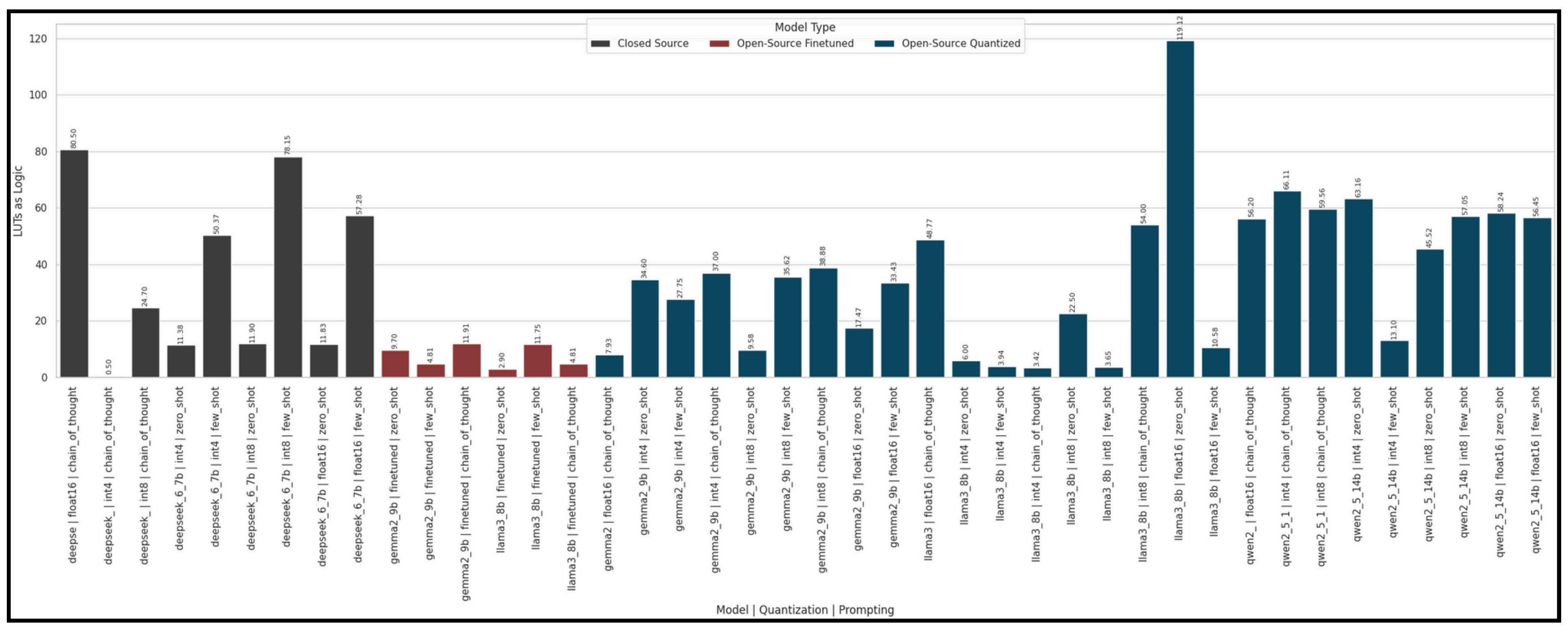
- Comparison of Total Power Consumption - Power Report



- Comparison of Datapath Delay - Timing Report



- Comparison of LUT as Logic Utilization - Utilization Report



Acknowledgement

We express our sincere gratitude for the opportunity, guidance, and support extended by Professor Joycee Mekie throughout the research. We are also deeply thankful to Mihir Agarwal and Zaqi Momin, whose previous work laid the foundation for this study. Despite having graduated, their continued guidance and insights were invaluable in shaping the direction and execution of this research.

References

- [1] M. Agarwal, Z. Momin, K. Prasad and J. Mekie, "VeriBench: Benchmarking Large Language Models for Verilog Code Generation and Design Synthesis," 2025 IEEE International Symposium on Circuits and Systems (ISCAS), London, United Kingdom, 2025, Available: https://doi.org/10.1109/ISCAS56072.2025.11044004.
- [2] DataCamp, "Zero-Shot Prompting: Getting Started," [Online]. Available: https://www.datacamp.com/tutorial/zero-shot-prompting
- [3] DataCamp, "Few-Shot Prompting: A Practical Guide," [Online]. Available: https://www.datacamp.com/tutorial/few-shot-prompting
- [4] Prompting Guide, "Chain-of-Thought Prompting," [Online]. Available: https://www.promptingguide.ai/techniques/cot
- [5] Intel, "Verilog HDL Example Designs," [Online]. Available: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-examples/horizontal/verilog.html
- [6] Hugging Face. [Online]. Available: https://huggingface.co/