Veritas Colab Lab Report

Name: Naveen Kumar Senthil Kumar

NetID: ns6503

This report documents the process and outcomes of experimenting with the Veritas Colab tutorial for generating CNF (Conjunctive Normal Form) equations for hardware designs. The designs tested were a 3-bit adder and a 3-bit multiplier. This study highlights the iterative refinement process of CNF generation prompts and the identification of logical inconsistencies despite syntactic correctness.

The CNF generation was performed using the GPT-4o-mini model. While the model produced syntactically valid CNF expressions after prompt optimization, several logical deviations were identified when comparing expected and generated truth tables. The analysis emphasizes the importance of semantic validation alongside syntactic checks.

Iterative Process

Multiple iterations were required to obtain syntactically valid CNF outputs. Initial generations produced CNF expressions containing invalid characters such as backslashes, dashes, and underscores in variable names, which resulted in syntax parsing failures.

By progressively refining the prompts to restrict special characters and enforce consistent variable naming, the CNF generator produced clean, processable files. These could be used for automated verification via the tutorial scripts, confirming syntactic correctness.

However, subsequent truth table comparison revealed that the generated CNFs—though valid in structure—did not always align with the true logical behavior of the circuits.

Observed Errors

Even though the outputs passed preliminary parsing and I/O script tests, logical mismatches were detected during truth table verification. The primary issue observed was incorrect logical mapping between input combinations and their expected CNF-derived outputs.

Wrong Behavior Examples

1. Adder (3-bit)
Input: A=000, B=000, Cin=0

Expected Output: Sum=000, Cout=0

Generated CNF Output: Sum=010, Cout=1

Analysis: The CNF erroneously asserts a carry and drives the sum to a nonzero value even when both inputs are zero. This demonstrates that although the CNF structure is syntactically valid, the underlying boolean relationships are incorrect.

2. Multiplier (3-bit)
Input: A=000, B=000

Expected Output: Product=000000

Generated CNF Output: Product=111111

Analysis: Instead of producing an all-zero product for zero operands, the CNF drives all output bits high. This reflects a total inversion of logical behavior, indicating severe semantic misrepresentation.

Lessons Learned

1. Prompt refinement successfully fixes syntax but does not guarantee semantic correctness.
2. Passing basic script checks can be misleading; logical correctness requires complete truth table validation.
3. Full truth table comparison is essential to reveal deeper logical inconsistencies.
4. Formal verification or equivalence checking should complement CNF generation to ensure logical soundness.

Conclusion

The Veritas Colab experiments with the adder and multiplier modules demonstrated that syntactic success does not imply logical fidelity. Although the refined prompts allowed generation of clean CNF files that could be processed by verification scripts, the outputs failed to reflect the correct circuit behavior.

This outcome underlines the need for stronger semantic validation, either through exhaustive simulation, formal methods, or model-level improvements, to ensure that generated CNF equations faithfully represent the intended hardware functionality.