Name : Dhirajzen Bagawath Geetha Kumaravel
NetID: db5309

Veritas Colab Lab Report

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, subtractor, and multiplier. The study highlights both the iterative process of refining CNF generation prompts and the discovery of logical errors despite apparent syntactic correctness. The CNF generation was performed using the **GPT-4o-mini** model. Although it produced syntactically valid CNF after prompt refinement, its limitations in logical reasoning contributed to incorrect truth table behaviours. The choice of model is therefore highlighted as a significant factor in the outcomes.

Iterative Process

Multiple iterations were required to generate usable CNF outputs. Initially, the CNF equations produced contained unwanted backslashes, dashes, and underscores in variable names, which led to syntax issues. By adjusting the prompts to explicitly forbid these symbols, syntactically valid CNF outputs were eventually produced. This allowed the designs to pass script-based input-output tests.

Observed Errors

Despite passing basic script tests, the generated CNF equations were not logically correct. A comparison with full truth tables revealed that while certain inputs produced correct outputs, the underlying logical relationships were wrong. This mismatch indicates that CNF correctness was superficial, with deeper logical flaws remaining.

Wrong Behaviour Examples

1. **Adder (3-bit):**
   Input: A=000, B=000, Cin=0
   Expected: Sum=000, Cout=0
   Generated CNF Output: Sum=010, Cout=1

   - This indicates the CNF incorrectly asserts carry and sets a nonzero sum even when no inputs are active.

2. **Subtractor (3-bit):**
   Input: A=000, B=000, Din=0
   Expected: Difference=000, Borrow=0
   Generated CNF Output: Difference=100, Borrow=1

   - The CNF falsely introduces a borrow and a nonzero difference when subtracting zero from zero.

Name : Dhirajzen Bagawath Geetha Kumaravel
NetID: db5309

3. **Multiplier (3-bit):**

    Input: A=000, B=000

    Expected: Product=000000

    Generated CNF Output: Product=111111

    - Instead of yielding zero, the CNF erroneously drives all outputs high, indicating a completely incorrect logical mapping.

Lessons Learned

1. Prompt refinement can correct syntax but not semantics.
2. Passing selective script-based verification does not imply functional correctness.
3. Full truth table validation is essential to uncover hidden logical flaws.
4. Generated CNF equations must be cross-verified with known good truth tables or formal verification tools.

Conclusion

The Veritas Colab experiments revealed that while syntactically correct CNF equations can be generated through iterative prompt refinement, logical accuracy remains a major limitation. The adder, subtractor, and multiplier case studies showed clear examples where the CNF passed tests but violated the expected truth tables. Stronger semantic validation is required for Veritas to reliably produce correct hardware representations.