Homework Assignment 5
Total Points: 50

Professor Karem A. Sakallah

EECS 598-002: Formal Verification of Hardware & Software Systems

Assigned: March 5, 2024
Due: March 12, 2023

Guidelines

- The College of Engineering Honor Code applies to all work in this course.
- The due date is firm. Follow submission instructions (at the end).

Objectives

This assignment explores the use of the AVR verifier to check the safety of state transition systems.

1 [Checking Sequential Equivalence] (20 Points)

The Verilog file SeqEqvMiter.v specifies a miter that checks the equivalence of two single-input single-output sequential circuits. The circuits are encoded as modules that are instantiated on lines 8 and 9 in the miter.

1.1 [Checking Equivalence in "IC3" Mode] (5 Points)

Run AVR on SeqEqvMiter.v and report the following:

- The size of the (miter's) state space (total number of states)
- The number of framees explored by AVR
- The frame number at which convergence was established
- The total number of SAT calls during AVR's run
- The size of the inductive invariant that proves equivalence

1.2 [Exploring Non-Equivalent Variants] (10 Points)

Now suppose that you are told that Circuit 2 is actually a re-implementation of Circuit 1 using a one-hot state encoding and that its output Z becomes true in exactly one of these states. In Which state of Circuit 2 should Z become true in order to yield the longest counterexample demonstrating non-equivalence? Can you infer from this experiment the sequential depth of the miter?

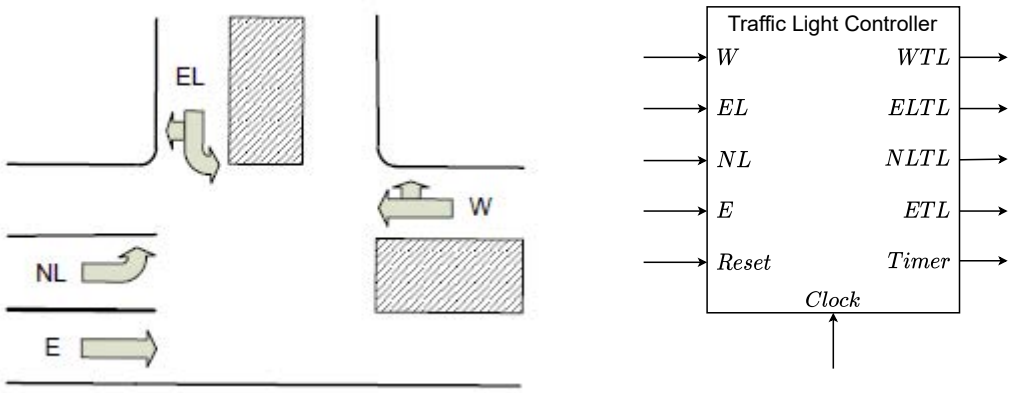

Figure 1: Traffic intersection and the input/output interface of its traffic light controller. The controller inputs W, EL, NL and E are 1-bit sensors that indicate the presence or absence of cars in the indicated traffic lanes. Each sensor has a corresponding output with a TL (traffic light) suffix that causes its associated traffic light (not shown) to cycle between Green, Yellow, and Red. The controller also has a timer output to limit the amount of time that a signal can be Green.

1.3 [Checking Equivalence in bmc Mode] (5 Points)

Repeat **1.2** by running AVR in bmc mode with a suitable completion threshold. Justify the particular threshold you chose.

2 [Specifying and Checking Safety Properties] (30 Points)

This problem mimics a typical verification scenario. The Verilog file TLC.v specifies the controller of the traffic signals for the intersection shown in Figure 1. The specification *is buggy* and your task is to identify and correct the bug(s) and produce a certificate of safe operation.

The deliverables for this problem are:

- (15 Points) The code snippet that must be added to TLC.v to specify and check for safe operation.
- (2.5 Points) Points) The witness file that demonstrates unsafe operation.
- (10 Points) The cause of unsafe operation and the required modification to TLC.v to eliminate it.
- (2.5 Points) The proof.smt2 file that serves as a certificate of safe operation.

You may find it helpful to refer to Figure 2 which shows the state transition graph corresponding to the Verilog code.

Submission Instructions

1. Create a directory named <your uniquname>_hw5 containing all of your work. This should include all files created by you as well as any requested files generated by running AVR. For each AVR run, capture the terminal log in a text file with a suitable name and include these

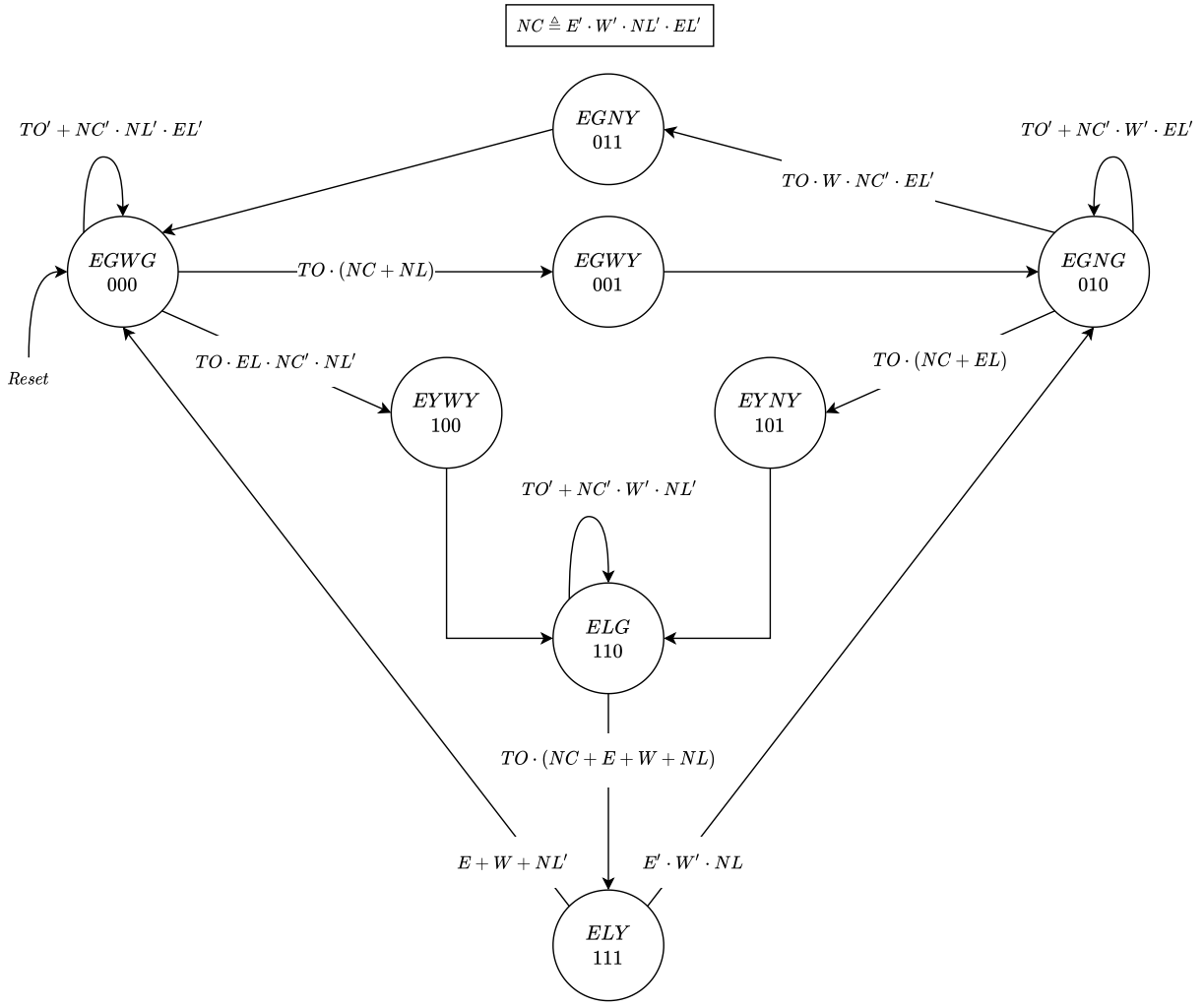

Figure 2: State Transition Graph corresponding to the Verilog code in TLC.v

files in your submission. The narrative parts of your submission should be typed and saved as PDF files. Handwritten and scanned documents or non-PDF documents ARE UNACCEPTABLE AND WILL NOT BE GRADED.

2. Zip the entire directory using "zip -r <your uniquename>_hw5.zip <your uniquname>_hw5" and upload to Canvas.