ECE 522/622 - Spring 2025
Modeling and Verification of Embedded Systems
Lab #4: Open Ended

General instructions:

1. The objective of this lab is to apply knowledge from the course to a new problem of your own choosing, and to get practice working without explicit step-by-step instructions.
2. The lab is intentionally open-ended. The options below list ideas that give a sense of appropriate scope for a lab 4 project. You can pursue one of these ideas, or an entirely different idea that interests you.
3. Choose a problem that is interesting but simple enough to complete in a few weeks. You may not know in advance if your project will succeed. That is ok. Good quality work that reaches a negative result will still earn a high score.
4. Lab reports should be typed, with name and ID at the beginning of the lab report. Include your model files as part of your submission. You can work with a partner. If you do so, make a single submission with both names on it.
5. Lab 4 is due on 5/16. It is required for ECE622 students, but optional for ECE522 students. ECE522 students that will do lab 4 should email me by 5/2 to declare. If you have a proposed project idea, and want feedback about it, email me by 5/2 so that we can arrange a meeting to discuss. I'll be happy to give feedback on your proposed direction and about whether it may be too simple or too complicated.

Option A: Use a new tool or feature to analyze a familiar model.

If evaluating a new tool, try to use a model that is familiar and simple. Your goal could be to analyze a model using the new tool and also analyze the same model (although syntax may be different) using one of the tools you've learned in class (SAT-based reachability, timed automata in UPPAAL, hybrid systems in PHAVerLite). You can then compare the findings to make sure their analysis is consistent, and to add your commentary on the strengths and weakness of each method. Here are some examples:

- Model a simple system using the **PRISM** probabilistic model checker. Do probabilities of "0" and ">0" in PRISM correspond to "unreachable" and "reachable" in a non-probabilistic reachability check using SAT from Lab 1?
- Create simple models in the **Promela** language and then use the **SPIN** model checker to check reachability properties. For example, you could use one of the smaller designs from Lab 1 and then show that the new results match your findings from Lab 1.
- Try same as above bullet but using the **nuXmv** model checker instead of SPIN.
- Apply some advanced features of UPPAAL or PHAVerLite (e.g. forward-backward reachability in PHAVerLite).

Option B: Use a familiar tool with a new model.

If using a familiar tool to verify properties of a new model, you should make every reasonable simplification, so that your problem has only a few variables. The goal should be to demonstrate a sound approach, not a grand scale. Here are some examples:

- An SRAM cell has 2 state nodes and can be represented in PHAVerLite as a system with 2 variables. Try to prove, for some range of initial conditions, that the cell will not spontaneously flip its stored bit. In other words, that it cannot reach the part of the state space that corresponds to the opposite stored bit.
- A DRAM cell has a single state node that decays between periodic refreshes. Making reasonable assumptions about decay rate, prove in PHAVerLite that a DRAM cell retains state when refreshed at 64ms period.
- A UART receiver circuit synchronizes on the falling edge of a data line, and then counts clock cycles to determine when to sample the subsequent bits on the data line. Sampling errors can occur if the receiver clock is much faster or slower than it should be. Model this in UPPAAL or PHAVerLite and perform analysis to find the smallest amount of clock period error that can cause bits to be sampled incorrectly.
- Verify path delay bounds on a simple combinational circuit path using timed automata. Circuit path should have only a few gates. Check literature for prior approaches of using timed automata for circuit timing.
- Prove reachability or unreachability of a state on some FSM that you have encountered outside of this course. The main work for this project would likely be translating your FSM into the format accepted by your solver from lab 1.
- Modify your lab 1 code so that you create an inductive proof that certain states are unreachable. Induction can prove unreachability in a single step by showing that a state is unreachable from non-deterministic initial state.
- Use a hybrid systems verifier like PHAVerLite to model kinetic time dilation from special relativity and then analyze the twin paradox to show that twins can reach different ages due to their motion.