REPORT B07901075 李彥儒

(i)My implementation

(1)For building the bdd of the circuit, I first complete th function buildBdd. I use DFS order to construct the bdd, simply get the right and left bdd node and use “&” operation to build the new bdd node. And I build bdd from each output, inout and latch, thus the whole circuit is built.

(2)The initial state is build from a constant one bdd node, “and” each latch’s bdd node’s complement. Thus the initial state is all zero.

(3)The transition relation is built with transition function “xnor” with next state bdd. The transition function is actually the current state function, which is the input of latches. And the next state bdd can be gotten from calling variables with postfix “_ns”. Simply “xnor” them get the _tri. Conduct existential quantifier to get _tr.

(4)To buildPImage, simply follow the procedure of lecture note. “&” _tr and the _reachable state to compute the new reachable states. If there is no new reachable states, fix point is reached.

(5)For runPCheckProperty, simply “&” the reachable state and the monitor. If the result bdd is constant zero, the monitor is safe. If the monitor is violated, I use getCube to find a counter example and back trace using the transition relation _tri to get the inputs sequence that leads to the counter example.

(ii)My assertion in the abstracted vending_abs.v

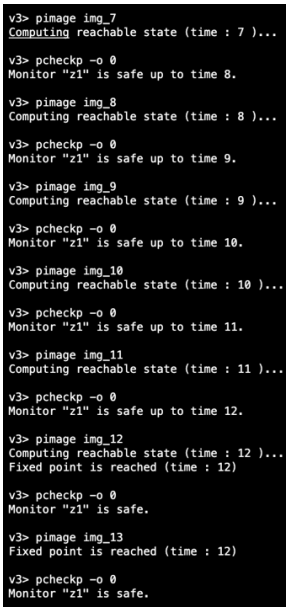
Monitor p : when the machine cannot make the change, it will output exchange different from the input value

Monitor q : check if the machine makes the correct exchange for ITEM_A

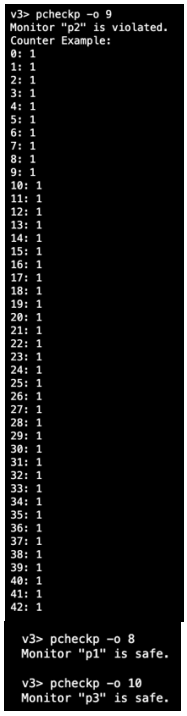
Monitor t : check if the input item is equal to the output item when the type out item is not ITEM_NONE

(iii)My verification results

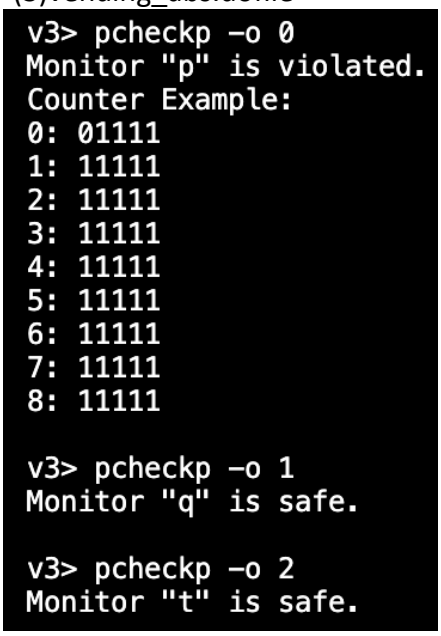
(1)a.dofile



(2)Traffic.dofile



(3)vending_abs.dofile

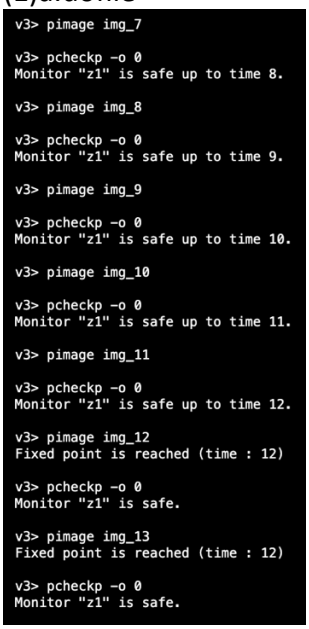


The verification results are essentially the same as the ref code given by the TA, and after checking by simulation, the results are as expected.

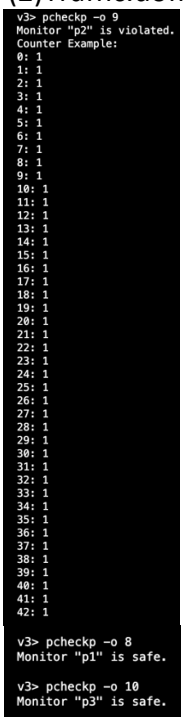
(iv) The comparison with the ref program

For ref program, the output is as follows

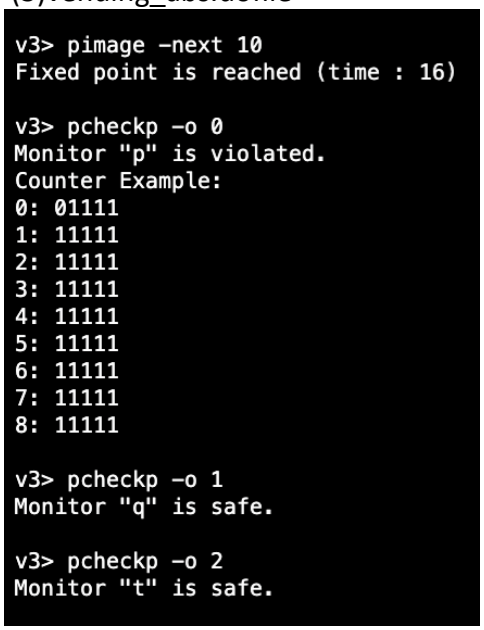
(1)a.dofile



(2)Traffic.dofile



(3)vending_abs.dofile



(v) the advanced techniques and/or abstraction of the design

I add restrict to improve the computation of image. And I also abstract the original vending.v to vending_abs.v, where there is only 1 and 5 dollars and ITEM_A. And the max number of 1 and 5 coin is 3 and 1 respectively. After the abstraction, I can still catch the bug monitored by p. And the monitors q and t are both safe still, which is consistent with my result in homework 1.

The performance of my code is about 3 times slower than the ref code. Yet, it is able to give the answer of the abstracted vending machine in about 3 minutes on DV lab's work station, which is an acceptable result.