CS254 Lab Assignment

Date: Mar 9, 2016

1. **Familiarization with `abc`:** We wish to model an 8-bit sequential circuit (some sort of 8-bit counter) using and-inverter graphs (AIGs) in the tool `abc` from University of California, Berkeley. We will use `BLIF` (Berkeley Logic Interface Format) as the input language for this. The tool, however, allows several other input formats as well. We then want to use a restricted subset of commands of `abc` to manipulate the internal representation (both as ROBDDs and AIGs).

   Use the following commands to understand (a subset of) the functionality of various relevant commands in `abc`.

   (a) `help`: shows the list of commands

   (b) `<command> -h`: shows help for `<command>`
   From now on, before using any command, you should first see the help menu for that command to figure out the best options to use for that command. This will be implicitly understood, and not explicitly stated below.

   (c) `read file_name`: reads a *logic* network with primary inputs, primary outputs, latch inputs and latch outputs, in general.

   (d) `print_stats`: prints statistics about the network (number of nodes, levels, etc.)
   From now on, you can use `print_stats` at any point to figure out how much optimized your network has become.

   (e) `show`: displays the logic network
   This is a good way to visualize the network, but if your network has become large, obviously this has its limitations. You may also want to kill the display windows that you no longer need – otherwise, each invocation of `show` will pop up a new display window.

   (f) `strash`: convert to AIG

   (g) `show`: displays the AIG

   (h) `balance; rewrite -z; refactor -z`: AIG optimization

   (i) `show`: displays the optimized AIG

   (j) `renode`: group AIG nodes and convert them to logic blocks

(k) `show`: displays the logic network

(l) `fraig`: Transform to functionally reduced AIG

(m) `show`: displays the logic network

(n) `fraig_sweep`: merge logic network nodes based on fraig-ing

(o) `show -g`: displays the optimized logic network with gate names

(p) `bdd`: converts function of each node to an ROBDD

(q) `show_bdd node_name`: displays the *local* ROBDD for a node in the logic network

(r) `collapse`: collapses the fan-in of each primary output and latch input, and generates ROBDDs for all outputs and latch inputs

(s) `show -g`: displays the collapsed network with gate (node) names

(t) `show_bdd node_name`: displays the *global* BDD for a named node in the logic network.

(u) `strash`: convert the design back to AIG

(v) `write_blif`: write out the optimized design in `BLIF`!

2. **Assignment using** `abc`: Modify the sequential circuit (some kind of counter) given in the input file `8bitUpDownCC.blif` to have the following behaviour:

   - The value saturates at 11001100. In other words, whenever the state 11001100 is reached, the sequential circuit remains stuck at this state.
   - The counter becomes an up-down counter with the following behaviour. When the input `inp` is 1, the counter counts up from 0 to 11001100, and thereafter it remains stuck at 11001100. When the input `inp` is 0, the counter counts down; on reaching 0 it wraps back to 11001100 and keeps downcounting.

   You must optimize your circuit (number of gates and levels) as much as possible using abc commands (preferably those above, but you are free to experiment with other commands as well), and show us what sequence of commands you used. You must clearly indicate to the TA what each command you are using does.