EE219B: Homework 8

Professor Robert K. Brayton

Due April 2 (Tuesday)*

1. Consider a network shown in Figure 1. The goal of this problem is to compute the compatible set of permissible functions for y_3, using the BDD-based technique for computing CSPFs.

   Given:

$$
\begin{aligned}
T &= \{y_4, y_5, y_6\} \\
V &= \{y_1\} \\
y_j &= y_3 \\
f_4^C &= y_1 + y_3 \\
f_5^C &= \overline{y_3} \\
f_6^C &= y_4 y_5 \\
\mathbf{y} &= \langle y_2, x_3 \rangle \\
\mathbf{x} &= \langle x_1, x_2, x_3 \rangle \\
\mathbf{z} &= \langle z \rangle \\
\mathbf{v} &= \langle y_1 \rangle
\end{aligned}
$$

   (a) Compute $g_V(\mathbf{x})$, $g_Z(\mathbf{x})$, $g_{fi}(\mathbf{x})$.

   (b) Compute $(\mathbf{y} \equiv g_{fi}(\mathbf{x}))$. Note that $\equiv$ translates to an XNOR. Since $\mathbf{y}$ has two components, you do the XNOR separately for each component, and then AND the results.

   (c) Compute $Z^C((\mathbf{v}, y_j), \mathbf{z})$.

   (d) Compute $Z^C((g_V(\mathbf{x}), y_j), \mathbf{z})$.

   (e) Compute $(\mathbf{z} \equiv g_Z(\mathbf{x}))$.

   (f) Compute $F_3^C((y_2, x_3), y_3) = \forall x_1, x_2, z[((\mathbf{y} \equiv g_{fi}(\mathbf{x})) \wedge (Z^C((g_V(\mathbf{x}), y_j), \mathbf{z}))) \Rightarrow (\mathbf{z} \equiv g_Z(\mathbf{x}))]$

*TA: Sriram Krishnan (krishnan@eecs.berkeley.edu), Office hours: M5-6.

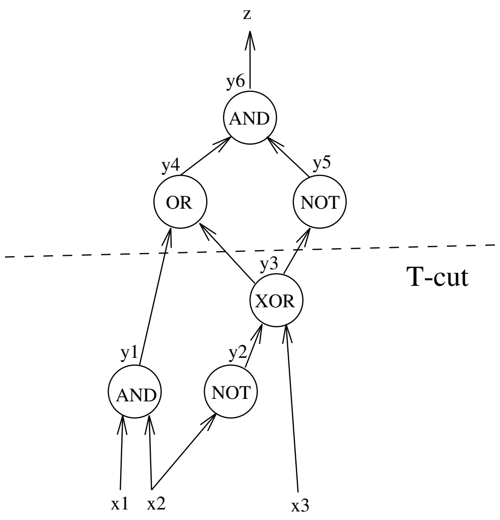

Figure 1: Circuit for T-cut problem

(g) By interpreting $F_3^C((y_2, x_3), y_3)$, describe which functions can be used at y_3.

2. This problem is intended to give you some experience programming in SIS. You are to write code to test if a given circuit is pipelined.

   We say a circuit is pipelined if there is no "path" from a latch back to itself, directly (i.e. the next state of the latch depends on it's present state) or indirectly (i.e. the next state of the latch depends on other latches which depend on it, directly or indirectly).

   Although a circuit may appear not to be pipelined structurally, it may be "functionally" pipelined, i.e. if the Boolean function at the latch input is computed, it is not found to depend on the latch output.

   (a) Write code to detect if a given circuit is pipelined. If it is not structurally pipelined, but functionally pipelined, you should be able to create an equivalent network that is structurally pipelined. You should implement a command called *network_test_pipeline* that should have three options:

      i. `-s` Should test if the previously read network is structurally pipelined. This should be the default option.

      ii. `-f` Should test if the network is functionally pipelined, and if so should write out an equivalent structurally pipelined network to a file whose name is supplied as the first argument (if no file name is given it should write out BLIF for the network to stdout).

iii. `-h` Should print the usage of the command.

The code should be well documented and should not core dump even if no network has been read in before invoking the command *network_test_pipeline*. The default option should be `-s`. The equivalent structurally pipelined network should pass the `verify_fsm` command, when invoked with the original network.

Refer to the *how to program in SIS* handout (page 10) for the path to a sample Makefile. You can implement your code within the test package. A directory containing examples called `examples` will be created in /projects/classes/290ls/ .

(b) Provide a print out of your code and conduct experiments using the benchmark circuits in /projects/classes/290ls/examples. Let us know where your source and executable files are, so that we can test them out.

3. (a) Write down the D-algorithm calculus for a 2-input OR gate.

   (b) Run the D-algorithm for the two circuits described in Figure2. Are the faults in the Figure detectable ?

4. Consider the circuit in Figure 3. Construct a product-of-sums formula which is satisfiable if and only if a stuck-at-0 of x is testable. Analyze the POS formula to see if the fault is testable. If testable, give an input test pattern. What is the output values of the good circuit and the faulty circuit for the test pattern?

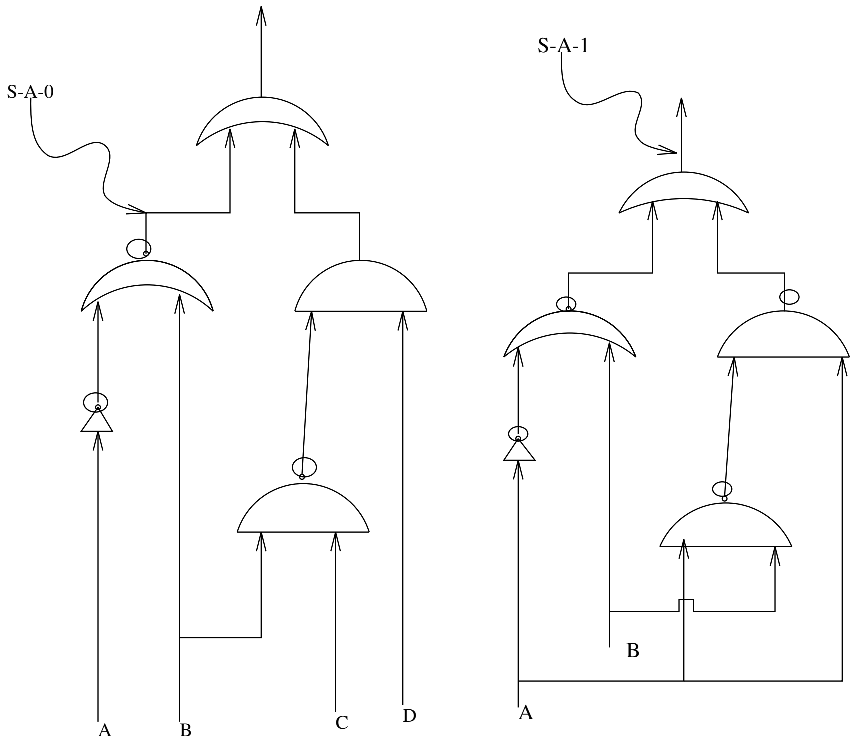

Figure 2: Circuits for the D-algorithm

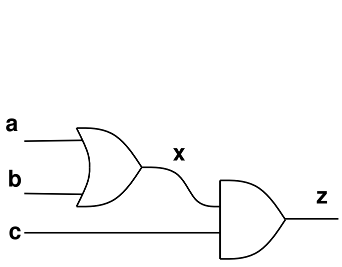

Figure 3: Circuit for the SAT problem