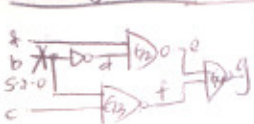


D Algorithm (example)



Step 0 All lines set to 1

① (step 1) excitation:
b = 1/0

② (step 2) implication:
d = 0/1

③ Frontier = { e_{12}, e_{13} }
(Step 0) propagates value since not at the end of values (ending pt). Since the algorithm doesn't know the goal value/type, you need to try different values, go through the pointer until reach contradiction, then go back and change value. This is the **decision tree** - keeps track of values and continue until successful.

④ (step 2) implication:
e = 1/0 ∴ write in decision tree
Note: don't know if good or bad decision
b = 1/0, d = 0/1

⑤ Frontier = { e_{13}, e_{14} }
(step 0) select c = 1 (arbitrary)
b = 1/0, d = 0/1

⑥ (step 2) implications: f = 0/1, g = 1
b = 1/0, d = 0/1

⑦ (step 4) back-track & make c = 0
⑧ (step 2) implication: f = 1, g = 0/1
b = 1/0, d = 0/1

⑨ No more implications ∴ (step 3) Primary output reached ∴ you have found a test for the diagram.
[abc = 110] tree marked as test found.

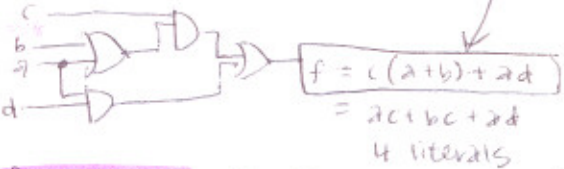
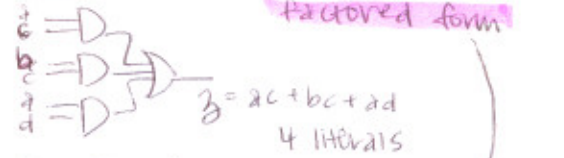
if nothing works & you switched, then the function returns that the fault is untestable

∴ (step x) is from D-Algorithm //

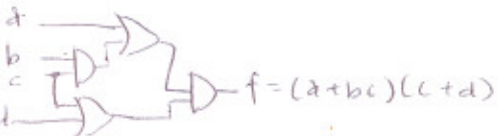
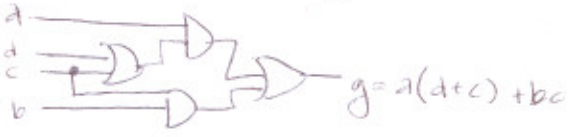
Multilevel synthesis (circuit logic)

literal - occurrence of variable or complement

Example



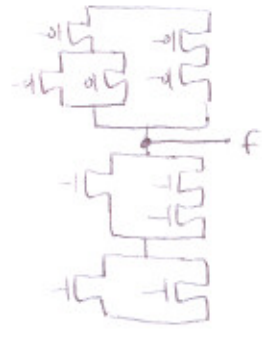
factored form - directly models circuit



How do you find out if it is optimal? you don't. ∴ BDDs help to show equivalency, but optimal

∴ we will use # of literals as a crude way to measure optimal-ness. Literals do not measure area, wire, etc. ∴ it is a CRUDE method.

For any expression, draw CMOS
* $f = (a+bc)(c+d)$



∴ (step x) is from D-Algorithm //