1) [ATPG: 15 points] For the circuit shown in Fig. 1:

![Circuit Diagram]

Fig. 1: Test for faults x/0, y/0 and \{x/0, y/0\}.

a) Find the set of all tests that detect single stuck-at fault x \textit{s-a-0}.
b) Find the set of all tests that detect single stuck-at fault y \textit{s-a-0}.
c) Find the set of all tests that detect multiple stuck-at fault \{x \textit{s-a-0}, y \textit{s-a-0}\}.
d) Use path sensitization to derive these tests.
e) Relate the results to the issue of “Test invalidation in presence of redundancies.”

2) [ATPG: 15 points] For the circuit shown in Fig. 2:

a) Find the set of all tests that detect single stuck-at fault b \textit{s-a-1}.
b) Find the set of all tests that distinguish between single stuck-at faults (i.e. their fault effects) \(\text{a \textit{s-a-0} and c \textit{s-a-0}}\).
c) Derive a test for the multi-fault \{a \textit{s-a-0}, b \textit{s-a-1}\}.
Fig. 2: Test for faults b/1, tests for distinguishing a/0 from c/0 and test for multi-fault \{a/0, b/1\}.

3) **[ATPG Checkpoints: 25 points]** For the circuit shown in Fig. 3, solve the following:

![Circuit Diagram]

Fig. 3: The circuit diagram related to Checkpoint faults

a) (10 pts) Suppose we wish to derive a set of tests to *distinguish* between all distinguishable single stuck-at faults in the circuit of Fig. 3. Identify a set of faults for which tests *need not* be derived.

b) (5 pts) List all the checkpoint faults of this circuit.

c) (10 pts) Find a *smallest/minimal* subset of checkpoint faults that must be targeted for test generation if the *detection* of all single stuck-at faults is the goal.

4) **[Stuck-at faults at fanout stems and branches: 20 points]** For the circuit shown in Fig. 3:

a) (15 pts) Derive a test that detects the following single faults (i) k/1; (ii) m/1; and (iii) n/1.
   You could use path-sensitization to derive these tests, or you could use the 'cec' command of ABC – your choice.
b) (5 pts) If any of the above faults is undetectable, remove the redundancy by removing redundant gates and/or lines.

5) [10 points] Let N be a combinational circuit composed only of NAND gates. Assume that all the primary inputs of the circuit have a fanout of exactly one (1). Show that any test set T that detects all single stuck-at-1 faults in the circuit, detects all single stuck-at-0 faults as well. Note: Do not assume that the circuit is fanout-free. Only the primary inputs (PIs) are fanout free. (A fanout of 1 means that the gate output is connected to only 1 other gate input, and this is also referred to as being fanout-free).

6) [10 points] Using the concept of Boolean differences, identify the conditions on the primary inputs of the circuit in Fig. 4 that disallow the changes in signal h to propagate to the output F. These conditions are called the observability don’t cares (ODC’s) of h. Using these ODC conditions, simplify the circuit. Recall, given a Boolean function $F(x, y, z, \ldots)$, the Boolean difference of $F$ w.r.t. $x$ is $\frac{\delta F}{\delta x} = F_x \oplus F_{x'}$, i.e. the XOR of cofactors. [You have to use Boolean differences to simplify the circuit. No points for K-map simplification. That’s trivial. This concept will help you understand how Boolean differences are related to fault propagation conditions as well as don’t cares.]

7) [5 points: Equivalence checking versus bug-detection in arithmetic circuits] On the class website, along with this HW, I have uploaded two BLIF files. They correspond to a 16-bit Mastro-
vito GF multiplier (MastrovitoF_q16.blif) and another 16-bit Montgomery GF multiplier (MontgomeryF_q16.blif). As described in my book chapter, (which we will study in the next few lectures) these architectures perform modulo-multiplication, but are based on different mathematical concepts; due to which these designs do not exhibit any internal structural or functional equivalences. As a result, SAT/AIG-based techniques are unable to prove equivalence between them. Instead of taking my word for it, you will gain a first-hand experience for yourself.

- Input the two designs into the ABC tool, and miter them.
- Using print_stats, strash, ifraig, print_stats, identify the structural similarity in the design. Let $N_1$ be the number of AIG nodes in the miter before fraiging, and $N_2$ be the number of AIG nodes after fraiging. Then $\frac{N_1 - N_2}{N_1}$ roughly depicts the structural similarity as a percentage.
- Solve sat on the miter to perform the combinational equivalence check. How many years does it take to prove equivalence of (fairly small) 16-bit datapath circuits?
- Now introduce a bug in any one of these circuits, by making any modification to any one of the BLIF files. Now run ABC-CEC and report whether bug catching or correctness proofs for arithmetic circuits is easier.