EECS 579 Homework No.2 Solutions

Problem 1 (10 points) *Functional equivalence* Text, Page 79, Problem 4.8.

Method 1: The faulty functions for the circuit of Figure 4.12 corresponding to the two faults are:

c s-a-0: b(ab)' = a'b

f s-a-1: (a + b)a' = a'b

These two functions are the same. Hence the two faults are equivalent or indistinguishable.

Method 2: You can also show the two faults are equivalent by showing they have exactly the same test set. This is easily done using any method to derive all possible tests for each fault. Both faults are detected by the singleton test set {ab = 01} and so are equivalent.

Problem 2 (10 points) *Initialization fault* Text, Page 79, Problem 4.2.

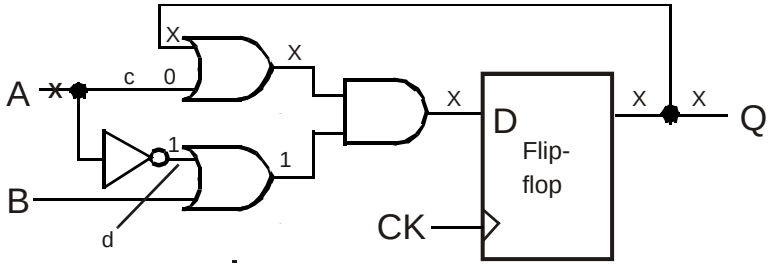


Fig. A

In the circuit of Figure 4.1 (copied in Fig. A above), let Qp denote the present state at the output Q of the flip-flop. Let the next state, i.e., the output of the AND gate, be Qn. We can write the circuit's next-state function as

Qn = (Qp + A)(A' + B)

Part 1: If we set A = 1, this function becomes Qn = B, which is independent of the present state. That is, irrespective of the present state, the next state can be set to a value, which is uniquely determined by primary inputs. This makes the fault-free circuit initializable. When the fault A s-a-0 is present, the above equation reduces to Qn = Qp. Thus, starting with Qp = X, the unknown value, Qn can never be changed to any value other than X. Therefore the circuit will remain uninitialized in the presence of A s-a-0.

Part 2: The last part of the problem is stated ambiguously. There are three reasonable interpretations, all of which were accepted.

Soln. 1. If "initialization" means "initialization to 0 or 1" (which is the usual interpretation used in the testing literature) then from the next-state expression we can easily determine that no other single stuck-at fault in the circuit will prevent initialization. For example, consider the s-a-0 fault on the top branch *c* of the fanout from primary input line A. The corresponding faulty next-state function is Qn = Qp(A' + B), which can be set to 0 when Qp = X by applying A = 1 and B = 0.

Soln. 2. If "initialization" means "initialization to 1" as in the Part 1 of the problem, then *c* s-a-0 (and several other faults) will prevent initialization to 1.

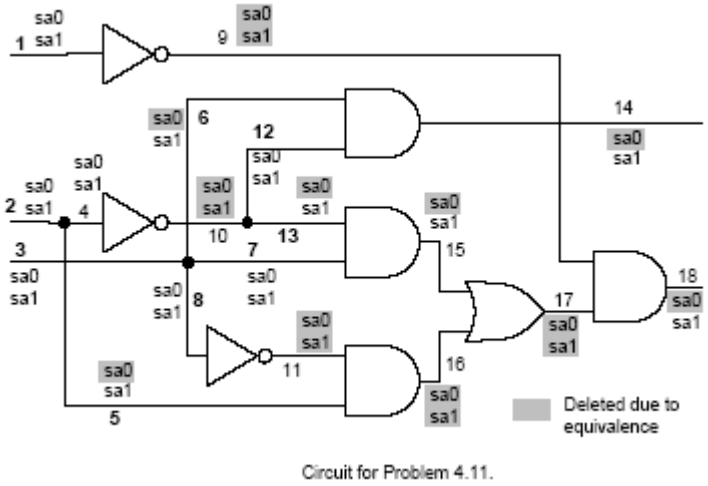
Soln. 3. Several of you noted that the clock line CK s-a-0 (inactive) will prevent any form of initialization. This renders the circuit completely non-operational and, of course, non-initializable.

Problem 3 (15 points) *Fault collapsing* Text, Page 80, Problem 4.11.

(a) The given circuit is shown in Fig. B below with the fault sites marked by numbers. The number of potential fault sites is 18

(b) Figure B shows all SSL faults, with the equivalent faults deleted by an output-to-input pass shaded. Of the original 36 faults, only 20 remain, giving a collapse ratio of 20/36 = 0.56.

.



Circuit for Problem 4.11.

Fig. B

(c) Following the approach given in the text, we start dominance fault collapsing with the checkpoint faults (which dominate the remaining faults). The checkpoints are marked by circled numbers in Fig. C; these are the three PIs and the seven fanout branches. Consequently, there are 10 checkpoints and 20 checkpoint faults. These 20 faults can be further reduced by the rules on p. 77. (Recall that two faults are equivalent iff they dominate each other.) The s-a-0 faults on lines 6 and 12 are equivalent, so either can be removed from the dominance fault set. Similarly, the s-a-0 faults on 7 and 13 are equivalent, as are 5 s-a-0 and 8 s-a-1. Thus the dominance fault set reduces to 17, giving a collapse ratio 17/36 = 0.47.

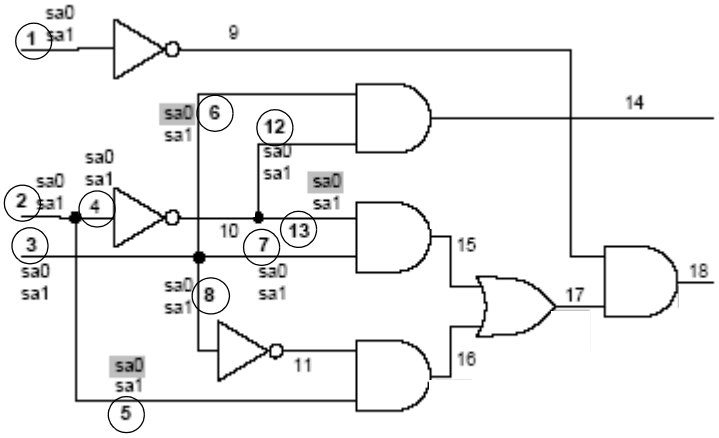


Fig. C

Note that the dominance fault set D is smaller than the equivalence fault set E obtained in Part (b), but only slightly. Also D can be obtained from E by deleting non-checkpoint faults from E. In general, neither D nor E is minimal, and may be further reduced by ad hoc methods.

Problem 4 (15 points) *Redundancy*

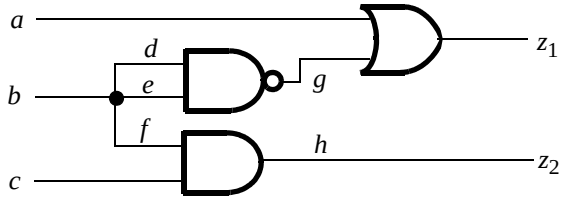


Fig. D

(a) The redundant faults are *d*/1 and *e*/1, which can be identified by inspection, or from the fault table given below, where they correspond to the only two columns with no x's.

(b) There are 6 checkpoints: the primary input lines *a*,*b*,*c*, and the fanout branches *d*,*e*,*f*. *Note*: Several people confused checkpoints, which are lines, with checkpoint faults.

(**c**) A straightforward approach is to construct a fault table for *C* and then examine the possible covers for the checkpoint faults; see below.The three tests {010, 101,111} shown shaded cover all the checkpoint faults (the first 12 columns) except the two undetectable cases. However, they fail to cover the non-checkpoint fault *g*/0, which is detected by 000 and 001.

Test *abc*	*a*/0	*a*/1	*b*/0	*b*/1	*c*/0	*c*/1	*d*/0	*d*/1	*e*/0	*e*/1	*f*/0	*f*/1	*g*/0	*g*/1	z_1/0	z_1/1	z_2/0	z_2/1
000				x									x		x			x
001				x								x	x		x			x
010		x	x			x	x		x					x		x		x
011		x	x		x		x		x		x			x		x	x	
100															x			x
101				x								x			x			x
110	x					x									x			x
111	x		x		x						x				x		x	

(d) This example implies that Checkpoint Theorem is only valid for non-redundant circuits. Although the fact that there are two primary outputs is a factor in making the given circuit a "counterexample", the Checkpoint Theorem is valid for multiple-output circuits, provided they are non-redundant.

The intuition behind this example is as follows: The s-a-0 fault *g* dominates the NAND gate's input faults *d*/1 and *e*/1, which happen to be on checkpoints. However *d* and *e* are duplicate lines so their s-a-1 faults are undetectable. So if we use the Checkpoint Theorem to delete *g*/0 from the fault set, we will not find a test for it. If we eliminate the redundancy, say by deleting line *d*, then *e*/1 becomes detectable by a test that will also detect *g*/0.

Note that a discussion of this issue appears on page 78 of the text.

Problem 5 (20 points) *Fanout-free circuits* Prove that all SSL faults in an *n*-input fanout-free circuit can be detected by at most $n + 1$ test vectors.

Method 1: As discussed in class, we can prove the theorem by induction on *m*, the number of logic levels or "depth" of the target circuit *N*. Without loss of generality, assume that all gates are NANDs (treating inverters as one-input NANDs) and that there are no inverters in the input/output lines of *N*. Then *N* has the structure shown in Fig. E. Observe that the *k* subcircuits denoted $N_1,N_2,...,N_k$ are all-NAND fanout-free circuits of depth $\leq m - 1$. This is a perfect set-up for a proof by mathematical induction on *m*.

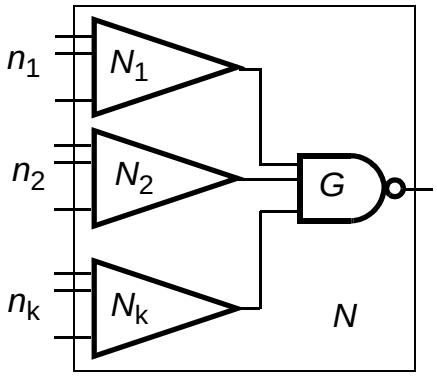


Fig. E

Basis step: When $m = 1$, *N* is just a single (NAND) gate. As we proved at the start of the course, *N* has a complete (and unique) set of $n + 1$ test vectors covering all its SSL faults. So the theorem holds true when $m = 1$.

Induction step: Assume that the theorem holds for all fanout-free circuits of depth $m - 1$ or less (This is the Induction Hypothesis). We must now prove that the theorem holds for the m-level case.

Each subcircuit N_i in Fig. E has depth $m - 1$ or less. Then by the Induction Hypothesis, N_i can be tested with $n_i + 1$ test vectors. Hence all k of these subcircuits can be tested with $\sum_{i=1}^{k}(n_i + 1) = n + k$ tests, by testing each N_i in sequence, and setting up the proper propagation conditions through G, namely by applying 1's to all inputs of G except the ith input (the one connected to N_i). Among these tests, at least k of them apply 1 to G, the output gate of N. Any k tests of this type can be applied in parallel, because the inputs of the N_is are independent and no fault masking can take place. This results in a set T of $n + 1$ tests that fully test each of $N_1, N_2, \ldots, N_k$.

It remains to show that T also detects any SSL faults in G. This is the case, because (a) G has the all-1s test applied to it as stated above, and (b) the tests for N_i must apply at least one 0 to G's ith input and 1's (the necessary propagating condition) to G's other inputs. Thus G has a complete set of $k + 1$ tests applied to it by T and so is fully tested. Hence T is a complete set of $n + 1$ tests for the m-level circuit N. (An alternative argument can be based on the fact that the faults in G dominate faults in $N_1, N_2, \ldots, N_k$.)

Finally we invoke the Induction Principle (which says that if the theorem is true for $m = 1$, it's true for $m = 2$, hence it's true for $m = 3$, and so on for all m) to finish the proof

Method 2: We can also prove the theorem by induction on n, the number of inputs of N, as several of you attempted. The proof is similar, but in this case, N is decomposed along the lines of Fig. F, where $n = K + k$ and $k \geq 2$. The induction hypothesis implies that the theorem holds for N^* with $K + 2$ tests. Then the problem is to merge the $K + 2$ tests for N^* with the $k + 1$ tests for G to test the entire circuit with $n + 1$ tests. The details are left as a exercise.

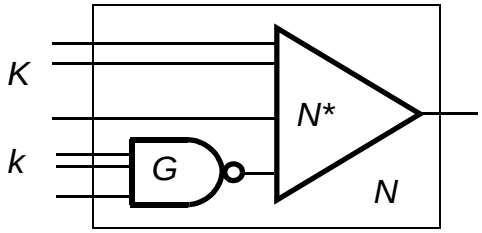


Fig. F

Problem 6 (10 points) *Combinational ATPG* D-Algorithm

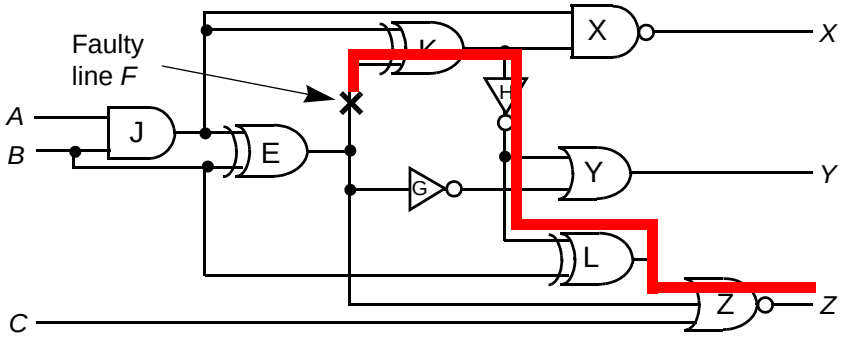


Fig. G

Decisions	**Implications**	**Explanatory Comments**
E=0	F=D',G=1,Y=1	Activate fault F/1. Note that output Y is blocked.
J=0	K=D',H=D,X=1	Propagate error through K. The D frontier is now at H.
B=0	L=D	Propagate error through L.
C=0	Z=D'	Propagate error through Z. End of D drive.
A=0 or 1		Justify J. The resulting test is ABC = 000 or 100.
		The sensitized path is shown by the heavy (red) line in Fig. G.

Notes: Instead of J=0 we can make the alternative decision J=1 to propagate the error through XOR gate K and eventually to output X. This results in the alternative solutions ABC = 110 or 111.

Also note that to make F=D', we set E=0. This implies that G=1 not G=D, since an error does not propagate "backwards" from gate K to gate G.

Problem 7 (20 points)

(a) Text, Page 207, Problem 7.4 *DALG applied to Fig. 7.39 h/0*

Decisions	**Implications**	**Explanatory Comments**
g=1	h=D,Y=1	Activate fault h/0.
f=1	k=D	Propagate error through k.
B=0	Z=D,d=0,e=1	Propagate error through Z. End of D drive.
	C=1	Implied by B=0 and f=1
A=0 or 1		Justify d. Test is ABC = 001 or 101.
		(The alternative decision B=1 gives the tests 110 and 010)

(b) Text, Page 207, Problem 7.5 *PODEM applied to Fig. 7.39 h/1*

Since we didn't cover SCOAP, it's OK not to use it here (It doesn't really help anyway.) We'll select inputs in lexicographic order when there is a choice.

Decisions	**Implications**	**Explanatory Comments**
		Start with ABC = XXX
		Objective=(g,0). Backtrace via d to get (A,0).
A=0	d=0	
		Objective=(g,0). Backtrace via e to get (B,1).
B=1		Objective=(g,0). Backtrace via e to get (C,1).
C=1	e=0,g=0,Y=0,h=D',f=0,k=0,...	Conflict: D frontier is empty. Backtrack.
C=0	e=1,g=1,h=1,...	Conflict: D frontier is empty. Backtrack.
	Now ABC = 01X and the subtree in heavy (red) lines in Fig. H has been traversed.	
B=0	e=0,g=0,Y=0,h=D',f=0,k=0,..	Conflict: D frontier is empty. Backtrack.
A=1		Objective=(g,0). Backtrace via d to get (B,0).
B=0	d=0,e=1,g=1,h=1,...	Conflict: D frontier is empty. Backtrack.
B=1	d=1,e=1,g=1,h=1,...	Conflict: D frontier is empty. All non-conflicting
	possibilities have now been tried, so PODEM concludes that fault h/1 is undetectable	

Notes: If different input selection criteria are used, the no. of conflicts (backtracks) and the decisions made will vary. The above execution of PODEM corresponds to the following decision tree, which searches quite a lot of the ABC space of 8 possible tests. Big searches are typically required by undetectable faults.

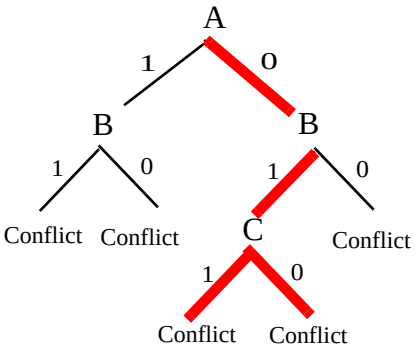


Fig. H