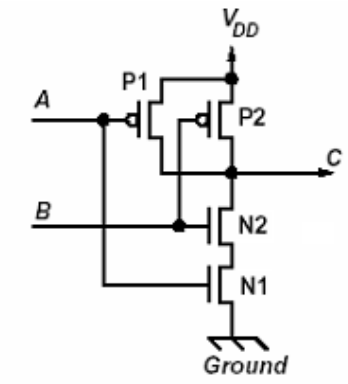
# ECE 538: VLSI System Testing
# Solutions to Homework 1

Krish Chakrabarty

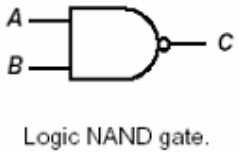
**Problem1:** The proof is very short and it is obtained by contradiction. Suppose there exists a fault u/d (d may be 0 or 1) in an irredundant circuit that complements the output. In that case, every input pattern is a test for u/d. This is true because the output is complemented in the presence of the fault, which implies that fault-free and faulty values will be different. Now, if every input pattern is a test for the fault, then every input pattern places the value d' on u. In other words, it is not possible to place the value d on u, i.e., the circuit is redundant. We can permanently set u to d'. In this way, we reach a contradiction.

**Problem 2:** The two faulty functions are indistinguishable (both fault functions are a'b), hence the faults are equivalent.

**Problem 3:** The circuit diagram is shown below. The table shows the test sequences for the stuck-open faults. Note that the stuck-open faults for N1 and N2 have the same tests, hence these faults are equivalent.



CMOS NAND gate.



Logic NAND gate.

| Test No. | Fault | Test: Vector 1, Vector 2 |
|---|---|---|
| 1 | P1 sop | 11,01 |
| 2 | P2 sop | 11,10 |
| 3 | N1 sop | 01,11 or 10,11 or 00,11 |
| 4 | N2 sop | 01,11 or 10,11 or 00,11 |

The following sequence of four vectors contains one vector pair for each fault in the above table: 11, 01, 11, 10. This sequence also detects all SSL faults in the logic model of the NAND gate. The following table lists all the equivalences between stuck-open and SSL faults.

| Stuck-at fault | Equivalent transistor faults |
|---|---|
| A s-a-1 | N1-ssh and P1-sop |
| B s-a-1 | N2-ssh and P2-sop |
| C s-a-1 | (P1-ssh or P2-ssh) and (N1-sop or N2-sop) |
| A s-a-0 | N1-sop and P1-ssh |
| B s-a-0 | N2-sop and P2-ssh |
| C s-a-0 | N1-ssh, N2-ssh, P1-sop, P2-sop |

**Problem 4:**
(a) $Z_{f1} = (a + c)'$

| a | b | c | $Z_{\text{fault-free}}/Z_{\text{f1}}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1/1 |
| 0 | 0 | 1 | 1/0 |
| 0 | 1 | 0 | 1/1 |
| 0 | 1 | 1 | 0/0 |
| 1 | 0 | 0 | 0/0 |
| 1 | 0 | 1 | 0/0 |
| 1 | 1 | 0 | 0/0 |
| 1 | 1 | 1 | 0/0 |

This physical fault is modeled by logical fault b: s-a-1.

(b) $Z_{f2} = a'$

| a | b | c | $Z_{\text{fault-free}}/Z_{\text{f2}}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1/1 |
| 0 | 0 | 1 | 1/1 |
| 0 | 1 | 0 | 1/1 |
| 0 | 1 | 1 | 0/1 |
| 1 | 0 | 0 | 0/0 |
| 1 | 0 | 1 | 0/0 |
| 1 | 1 | 0 | 0/0 |
| 1 | 1 | 1 | 0/0 |

This physical fault is modeled by logical fault b: s-a-0 or c: s-a-0.

**Problem 5:** Two patterns are required. The input combinations 0000, 0010, and 0001 can charge the output. To discharge it, we need 0011.

Pattern 0010 $\rightarrow$ 0011 is robust, other two are not. Reason: Parasitic capacitance between Y and Z. For the two other patterns, then initialization pattern does not charge this capacitance.

**Problem 6:** X* is useful in simulations with uninitialized memory elements. Example: a flip-flop controlling the select signal of a multiplexer ($Z = AC + B\overline{C}$) with both data inputs at 1. Logic simulation with the 3-valued logic will result in $Z = X$, whereas using the new logic, we get $Z = 1$ (which makes sense: no matter which input is selected, the output should be 1). Please refer to Figure 5.8 in the textbook.

**Problem 7:** This is simple once you get a gut feeling for it. C is simply a three-input majority function, which takes the value 1 when two or more inputs are 1. There are many ways to implement it. One of the simplest is: $C = X_1X_2 + X_1X_3 + X_2X_3$