A-PDF Merger DEMO : Purchase from www.A-PDF.com to remove the watermark

Solutions to Problems
from "Essentials of Electronic Testing"
©M. L. Bushnell and V. D. Agrawal, 2002

February 10, 2006

Please Read This

This manual contains solutions to all problems that appear at the end of the chapters in the book. At the end of the manual we have included the solutions to problems we used for the examinations in the Spring 2002 course at Rutgers University, and Spring 2004 and Spring 2005 courses at Auburn University.

In spite of all the care taken to ensure accuracy, we caution the user that some answers may contain errors as it is the first release of this manual. We will appreciate if any errors or comments are forwarded to us by email: *vagrawal@eng.auburn.edu* or *bushnell@caip.rutgers.edu*.

This manual has been created as teaching material that accompanies the book. To preserve its effectiveness, it should not be distributed. If necessary, only a very small set of solutions can be copied for distribution in the class. Please do not pass your copy on to others and ask any one requesting it to contact the authors.

Teachers can also use the presentation slides for 31 lectures (or an alternative sequence of 23-lectures), based on the book and available at the following websites:

http://www.eng.auburn.edu/∼vagrawal/COURSE/lectures.html

http://www.caip.rutgers.edu/∼bushnell/rutgers.html

We hope the readers of our book, both teachers and students, will benefit from this work. We acknowledge the help from colleagues and students in completing this solution manual and the assistance of the University of Wisconsin-Madison in its initial distribution.

Chapter 1: Introduction

1.1 Chip testing

The events of Example 1.1 are redefined as follows:

PQ:	chip is good	P:	chip passes the test
FQ:	chip is bad	F:	chip fails the test

A 70% yield means, $Prob(PQ) = 0.7$ and $Prob(FQ) = 0.3$. Following the analysis of Example 1.1, $Prob(P) = 0.68$. Then,

$$\begin{aligned}\text{Defect level} &= \frac{\text{Bad chips that pass tests}}{\text{All chips that pass tests}} \\ &= Prob(FQ|P) \\ &= \frac{Prob(P|FQ)Prob(FQ)}{Prob(P)} \\ &= \frac{0.05 \times 0.3}{0.68} = 0.022\end{aligned}$$

The defect level is $22,000$ ppm (parts per million).

1.2 Chip testing

Let x denote the escape probability, $Prob(P|FQ)$. Referring to the formula derived in Problem 1.1, a defect level of 500 *ppm* means,

$$\frac{Prob(P|FQ)Prob(FQ)}{Prob(P)} = \frac{x \times 0.3}{0.95 \times 0.7 + x \times 0.3} = 0.0005$$

This gives,

$$x = \frac{0.0003325}{0.29985}$$

Next, we obtain,

$$\begin{aligned}\text{Defect coverage} &= Prob(F|FQ) = 1 - Prob(P|FQ) \\ &= 1 - x = 0.99889\end{aligned}$$

The required defect coverage is 99.889%. This represents the capability of the test in detecting the actual "defects" that occur and should not be confused with the "fault coverage," which is defined for the "single stuck-at" fault model.

1.3 Test cost

Assuming that one vector is applied per clock cycle during the digital test, the rate of test application is 200 million vectors per second. Therefore,

$$\text{Digital test time} = \frac{1000 \times 10^6}{200 \times 10^6} = 5\ s$$

Adding the analog test time, we get

$$\text{Total test time} = 1.5 + 5.0 = 6.5 \ s$$

The testing cost for a 500 MHz, 1,024 pin tester was obtained as 4.56 *cents* in Example 1.2 (see page 11 of the book.) Thus,

$$\text{Cost of testing a chip} = 6.5 \times 4.56 = 29.64 \ cents$$

The cost of testing bad chips should also be recovered from the price of good chips. Since the yield of good chips is 70%, we obtain

$$\text{Test cost in the price of a chip} = \frac{29.64}{0.7} \approx 42 \ cents$$

41.8 cents should be included as the cost of testing while figuring out the price of chips.

1.4 Test cost and self-test

Following Example 1.2 of the book (pp. 10-11), we obtain

$$\text{ATE purchase price} = \$1.2M + 256 \times \$3,000 = \$1.968M$$

Assuming a 20% per year linear rate of depreciation, a maintenance cost of 2% of the price, and an annual operating cost of $\$0.5M$,

$$\text{Running cost} = \$1.968M \times 0.2 + \$1.968M \times 0.02 + \$0.5M = \$932,960/year$$

$$\text{Testing cost} = \frac{\$932,960}{365 \times 24 \times 3600} = 2.96 \ cents/second$$

Testing cost of the self-test design is 2.96 cents per second, down from 4.56 cents per second calculated in Example 1.2

1.5 Test complexity

Consider a cube of side d. The number of transistors (N_t) is proportional to the volume d^3, and the number of pins (N_p) is proportional to the surface area $6d^2$. Thus, the Rent's rule for the cube can be expressed as,

$$N_p = K \times {N_t}^{2/3}$$

where K is a constant, which depends on such technology parameters as the minimum feature spacing. For simplicity, we will assume that this constant is the same for the flat and cubic chips. Following Example 1.3 (pp. 12-13 of book), we define the test complexity, TC, as transistors per pin, or $TC = N_t/N_p$. For the cube,

$$TC_{cube} = \frac{N_t}{N_p} = \frac{N_t}{K{N_t}^{2/3}} = \frac{1}{K}{N_t}^{1/3}$$

Using the Rent's rule for a flat chip (Equation 1.5 on page 13 of book), we obtain

$$TC_{square} = \frac{N_t}{K{N_t}^{1/2}} = \frac{1}{K}{N_t}^{1/2}$$

Therefore,

$$\frac{TC_{square}}{TC_{cube}} = {N_t}^{1/6}$$

This ratio of test complexities continues to increase as the number of transistors (N_t) on the VLSI device grows. For example, for $N_t = 1$ *million*, the square-chip test complexity is ten times greater than that of the cubic-device. **The test problem of the cubic configuration is less complex than that for the flat chip.**

Note: Although chips at present are not designed as three-dimensional objects, three-dimensional packages and interconnects are in use. An interested reader may see the article: H. Goldstein, "Packages Go Vertical," *IEEE Spectrum*, vol. 38, no. 8, pp. 46-51, August 2001. Recently, Matrix Semiconductor announced plans to produce a three-dimensional memory chip. See, "Adding a Third Dimension to Chips," *Computer*, vol. 35, no. 3, p. 29, March 2002.

Chapter 2: VLSI Testing Process and Test Equipment

2.1 Test types

To reduce the warranty and product liability costs, the manufacturer must adopt a thorough but cost-effective test plan. A low failure rate, which may be as low as 100 parts per million, means that among one million chips shipped by the manufacturer there should be no more than 100 defective chips. A suitable test strategy requires adjustments to tests as the production ramps up. A realistic plan is as follows:

- Initial production: The manufacturer uses parametric tests and vector tests, the latter with coverage in the 95-100% stuck-at fault range. For high-speed microprocessor chips, at-speed critical path tests are run. The chips should be subjected to burn-in test for infant mortality.
- Matured production: If burn-in failures are lower than the required defect level then that test is eliminated or reduced to a sample basis. Any field returns are re-tested by the manufacturing tests. If these pass then the manufacturing tests are augmented, when necessary, by customer-supplied tests.
- Test optimization: Tests are optimized to reduce the manufacturing cost. First, test sequences that fail a larger number of devices are moved to the beginning. Second, test sequences that do not fail any devices are dropped. Such modifications change the emphasis from detection of *modeled faults* to detection of actual *defects*.
- Process monitoring: Once the chip goes into high-volume production, the manufacturing process and the outgoing product (chips) should be monitored to keep any variations within *statistical limits*. This means that various parameters, such as metal resistivity, polysilicon conductivity, transistor parameters, etc., should be within their three-sigma range (*average* $\pm$ 3 $\times$ *standard deviation*). Any excursions outside such a range are immediately diagnosed and the causes remedied.

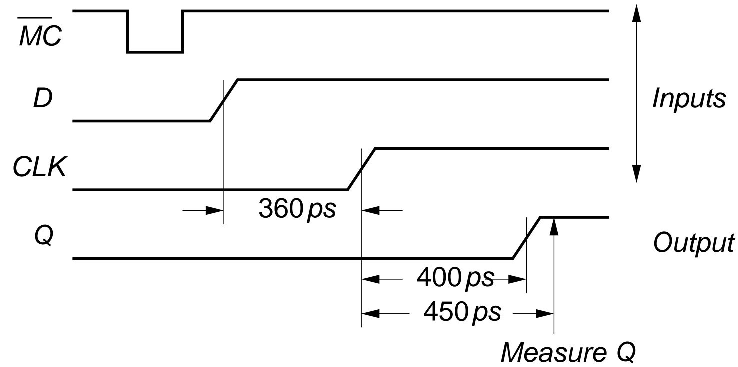
2.2 Contact test

Assume a diode drop of $0.7V$. Then, the pin voltage range for contact test is given by:

$$\begin{aligned}
\text{Upper range}: \; V_{pin} &= 0V - 0.7V - 100\mu A \times 2000\Omega \\
&= -0.9V \\
\text{Lower range}: \; V_{pin} &= 0V - 0.7V - 250\mu A \times 2000\Omega \\
&= -1.2V
\end{aligned}$$

2.3 Set-up time test

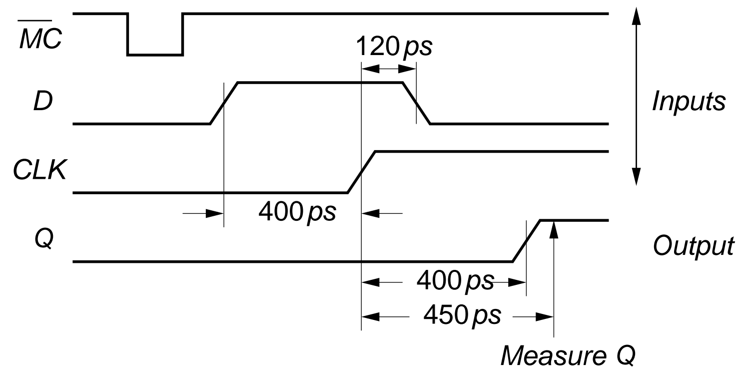
To test a set-up time, $t_{set-up} = 360ps$, apply the following waveforms to the chip (a clock-to-Q delay of $400ps$ is assumed):



At an interval of 450*ps* after the rising CLK edge, measure Q on the ATE. If $Q = 1$, the device passes, otherwise it fails. Using $\overline{MS}$ instead of $\overline{MC}$, repeat the above waveform sequence, but with D inverted and the expected Q signal also inverted. At an interval of 450μs after the rising CLK edge, again measure Q on the ATE. If $Q = 0$, the device passes, otherwise it fails. The same waveforms are applied simultaneously to all five D lines, and five simultaneous measurements are made on the five Q lines.

2.4 Hold time test

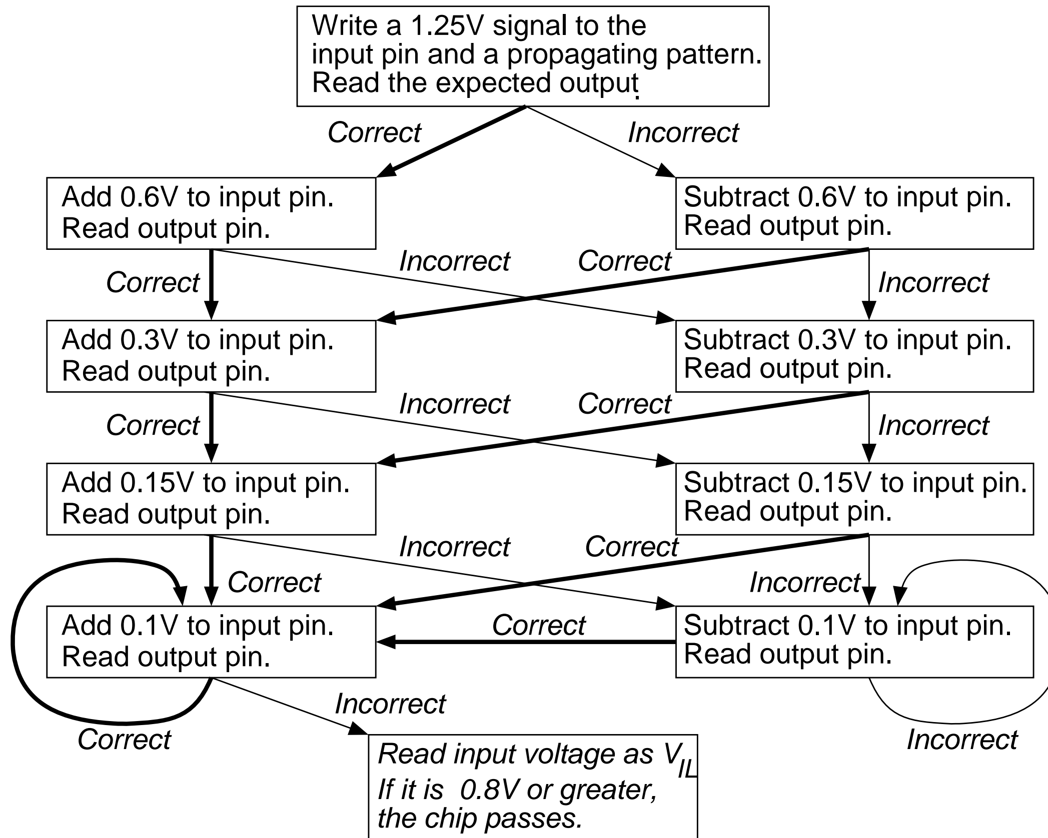
To test a hold time, $t_{hold} = 120ps$, apply the following waveforms to the chip (a clock-to-Q delay of 400*ps* is assumed):



At an interval of 120*ps* after the rising CLK edge, we lower the D line. If $Q = 1$ 450*ps* after the rising CLK edge, the device passes, otherwise it fails. Using $\overline{MS}$ instead of $\overline{MC}$, repeat the above waveform sequence, but with D inverted and the expected Q signal also inverted. At an interval of 450μs after the rising CLK edge, again measure Q on the ATE. If $Q = 0$, the device passes, otherwise it fails. The same waveforms are applied simultaneously to all five D lines, and five simultaneous measurements are made on the five Q lines.

2.5 Threshold test

Perform the threshold test as given on page 32 of the book, but with the following changes: Assume a $5V$ supply, and perform binary search to find V_{IL} and V_{IH}. The following procedure determines V_{IL}:



The advantage of this procedure is that it greatly speeds up the test. The test for V_{IH} is analogous.

Chapter 3: Test Economics and Product Quality

3.1 Economic decision

We start with the following formula for the price of the car deriven by John (Equation 3.2 on page 38 of the book):

$$P = 20,000 + \frac{20,000}{n} \quad dollars$$

where n is the number of breakdowns per 15,000 miles since John's car is driven 15,000 miles in a year. Because Laura drives only 5,000 miles per year, her car is expected to have $n/3$ breakdowns per year. Assuming a linear depreciation to zero value over 20 years and an average repair cost of \$250 per breakdown, the annual cost of driving is

$$\begin{aligned} C &= \frac{P}{20} + K + 250n/3 \quad dollars \\ &= 1,000 + \frac{1,000}{n} + K + 250n/3 \quad dollars \end{aligned}$$

where K is the cost of gasoline and regular maintenance, assumed to be the same for all models. To minimize this cost, we write

$$\frac{dC}{dn} = -\frac{1,000}{n^2} + \frac{250}{3} = 0 \quad \text{or} \quad n = \sqrt{12}$$

This is a minimum because $\frac{d^2C}{dn^2} > 0$. The price of a car for minimum transportation cost is,

$$P = 20,000 + \frac{20,000}{\sqrt{12}} = 25,774 \ dollars$$

Laura should invest in a car priced around 25,774 dollars.

3.2 Economic decision

(a) Let x be the daily wages of a technician and c be the cost of components on a board. When n technicians work in the assembly shop, the cost of one board is,

$$\begin{aligned} C(n) &= \frac{Warehouse\ cost}{n} + technician's\ wages + component\ cost \\ &\quad +workspace\ cost \\ &= \frac{10,000}{n} + x + c + \frac{500n^2}{n} \end{aligned}$$

To minimize this cost, we write

$$\frac{dC(n)}{dn} = -\frac{10,000}{n^2} + 1,000n = 0 \quad \text{or} \quad n = \sqrt{20} = 4.47$$

This is a minimum since $\frac{d^2C(n)}{dn^2} > 0$. We obtain the minimum cost as,

$$C(4) = C(5) = \$4,500 + x + c$$

To minimize the cost we should either hire four technicians, or reduce the workforce to five if more than five technicians were already employed.

(b) Substituting $x = 200$ and $c = 10,000$ in the last equation, we get

$$C(4 \text{ } or \text{ } 5) = \$4,500 + 200 + 10,000 = \$14,700$$

The minimum cost of a single-board system is $14,700.

3.3 Benefit-cost analysis

Please note a correction in the statement of this problem. The part (a) should read: Show that this scheme is beneficial for chips whose total cost is less than ten times the burn-in cost when the burn-in yield is 90%.

(a) *Complete elimination of burn-in:* Let C_t be the total cost of a chip in the present scheme where burn-in test is applied to every chip that passes the conventional test. Let C_b be the per chip cost of burn-in. C_t includes C_b, as well as another component, C_f, which accounts for the costs of fabrication, conventional test, etc. It is given by,

$$C_t = \frac{C_f + y_c C_b}{y_c y_b}$$

where y_c is the yield with the conventional test and y_b is the yield reduction due to burn-in. Since the cost of I_{DDQ} test is 10% of the burn-in cost and there is a 10% yield loss, the cost of a chip when burn-in is replaced by I_{DDQ} test is given by,

$$C'_t = \frac{C_f + 0.1 y_c C_b}{0.9 y_c y_b}$$

For the new scheme to be beneficial, we must have

$$C'_t < C_t \quad \text{or} \quad C_t < \frac{9C_b}{y_b}$$

For the given 90% burn-in yield, $y_b = 0.9$, and $C_t < 10C_b$. **The total cost should not exceed ten times the burn-in cost.**

(b) *Apply burn-in test only to chips that fail I_{DDQ} test:* Let y_b be the burn-in yield. Consider all chips that have passed pre-burn-in tests. A fraction y_b of these is "good" chips. We apply I_{DDQ} test to all chips passing the pre-burn-in test. Due to the 10% yield loss, this will produce a fraction $0.9y_b$ consisting of good chips. The remaining fraction, $1 - 0.9y_b$, must be subjected to the burn-in test to recover the lost yield. For the new scheme to be beneficial, we must have

$$0.1C_b + (1 - 0.9y_b)C_b < C_b \quad \text{or} \quad y_b > \frac{1}{9}$$

Burn-in yield should be greater than 1/9 or 11.1%.

3.4 Yield and cost

Let C_w be the cost of processing a wafer having N chips and let $y(A)$ be the yield of chips, where A is the chip area. Then the cost per good chip is obtained as,

$$C_c = \frac{C_w}{Ny(A)}$$

DFT changes the chip area to $(1+\Delta)A$. The number of chips on a wafer of area NA is now given by, $NA/(A+\Delta A) = N/(1+\Delta)$. The cost of a good chip with DFT is given by,

$$C_c(DFT) = \frac{C_w}{\frac{N}{1+\Delta}\; y(A+\Delta A)}$$

Therefore, the cost increase due to DFT is,

$$\begin{aligned}\text{Cost increase} &= \frac{C_c(DFT) - C_c}{C_c} \times 100 \;\; percent \\ &= \left[\frac{(1+\Delta)y(A)}{y(A+\Delta A)} - 1\right] \times 100 \;\; percent\end{aligned}$$

Using the yield formula of Equation 3.12 (p. 46 in the book), we get

$$\begin{aligned}\text{Cost increase} &= \left[(1+\Delta)\frac{(1+Ad/\alpha)^{-\alpha}}{(1+(1+\Delta)Ad/\alpha)^{-\alpha}}\right] \times \;\; percent \\ &= \left[(1+\Delta)\left(1+\frac{Ad\Delta}{\alpha+Ad}\right)^{\alpha} - 1\right] \times 100 \;\; percent\end{aligned}$$

which is the required result.

For the given data, $d = 1.25\ defects/cm^2$, $\alpha = 0.5$, $\Delta = 0.1$, and $A = 1\ cm^2$, we obtain

$$\begin{aligned}\text{Cost increase} &= \left[1.1\left(1+\frac{1.25\times 0.1}{0.5\times 1.25}\right)^{0.5} - 1\right] \times 100 \;\; percent \\ &= 13.86\%\end{aligned}$$

There is a 13.86% increase in the chip cost due to DFT.

3.5 Defect level and fault coverage

Defect level, DL, is given by Equation 3.20 (p. 50 of the book), as follows:

$$DL = 1 - \left(\frac{\beta + TAf}{\beta + Af}\right)^{\beta}$$

where T is the fault coverage, Af is the average number of faults on a chip of area A, and β is a fault clustering parameter. Further manipulation of this equation

leads to the following result:

$$\begin{aligned} (1-DL)^{1/\beta} &= \frac{\beta + TAf}{\beta + Af} \\ \text{or} \qquad T &= \frac{(\beta + Af)(1-DL)^{1/\beta} - \beta}{Af} \times 100 \;\; \textit{percent} \end{aligned}$$

which is the required result.

3.6 Defect level and fault coverage

Substituting the given fault density, $f = 1.45\ faults/cm^2$, the fault clustering parameter, $\beta = 0.11$, and the fault coverage, $T = 0.95$, in Equation 3.20 (page 50 of the book), we obtain the defect level as,

$$\begin{aligned} DL(T) &= 1 - \left(\frac{\beta + TAf}{\beta + Af}\right)^{\beta} \\ &= 1 - \left(\frac{0.11 + 0.95 \times 1.0 \times 1.45}{0.11 + 1.0 \times 1.45}\right)^{0.11} \\ &= 0.00522 \;\text{ or }\; 5,220 \text{ parts per million} \end{aligned}$$

The defect level is 5,220 parts per million (ppm).

(a) To obtain the fault coverage T for a required defect level of 1,000 ppm, we substitute $DL = 0.001$ in the formula derived in Problem 3.5. Thus,

$$T = \frac{(0.11 + 1.45) \times 0.999^{1/0.11} - 0.11}{1.45} \times 100 = 0.990$$

The required fault coverage is 99%.

(b) For a defect level of 500 ppm ($DL = 0.0005$), we get

$$T = \frac{(0.11 + 1.45) \times 0.9995^{1/0.11} - 0.11}{1.45} \times 100 = 0.995$$

The required fault coverage is 99.5%.

3.7 Defect level

Defect level, $DL(T)$, given by Equation 3.20 (p. 50 of the book), can be written as:

$$\begin{aligned} DL(T) &= 1 - \frac{(1 + TAf/\beta)^{\beta}}{(1 + Af/\beta)^{\beta}} \\ &= 1 - \frac{e^{TAf}}{e^{Af}} = 1 - e^{-Af(1-T)}, \quad \text{as} \quad \beta \rightarrow \infty \end{aligned}$$

Also, as $\beta \rightarrow \infty$, Equation 3.19 (p. 50 of the book) gives the yield,

$$Y = \left(1 + \frac{Af}{\beta}\right)^{-\beta} = e^{-Af}$$

Substituting this expression for yield in the defect level, we get

$$DL(T) = 1 - (e^{-Af})^{1-T} = 1 - Y^{1-T}$$

which is the required result.

Chapter 4: Fault Modeling

4.1 Boolean functions

An n-variable Boolean function is completely specified by its truth-table. The output column in this table is a 2^n-bit vector that can be in 2^{2^n} distinct states, each specifying a different Boolean function.

4.2 Initialization faults

In the circuit of Figure 4.1 (p. 62 of the book), let Q_p denote the present state at the output of the FF. Let the next state, i.e., the output of the AND gate, be Q_n. We can write the *next state function*, as

$$Q_n = (Q_p + A)(\overline{A} + B)$$

If we set $A = 1$, the next state function, $Q_n = B$, becomes 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 $Q_n = Q_p$. Thus, starting with $Q_p = X$, Q_n can never be changed to any value other than X and, therefore, **the circuit will remain uninitialized in the presence of this fault.**

Using the next-state expression, we can easily determine that no other single stuck-at fault in this circuit will prevent initialization. For example, consider the s-a-0 fault on the top branch of the fanout of A. The faulty next state function is $Q_n = Q_p(\overline{A} + B)$, which can be set to 0, when $Q_p = X$, by applying $A = 1,\ B = 0$.

4.3 Fault counting

See Section 4.5 (last paragraph on p. 70 of the book.)

4.4 Fault counting

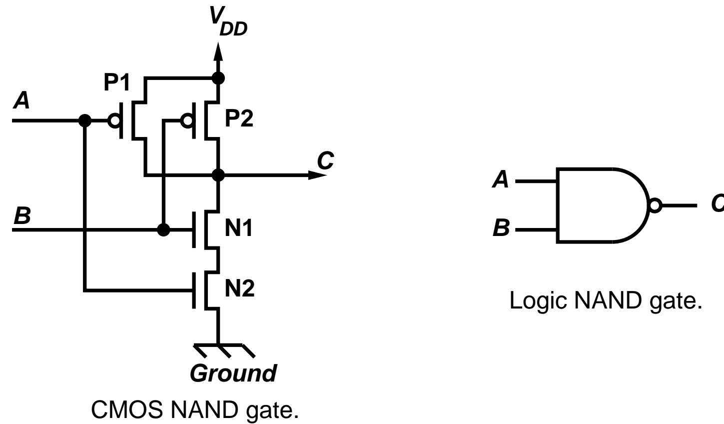
For the circuit of Figure 4.6 (p. 72 of book), we have

$$\begin{aligned}\text{Number of fault sites} &= \text{PIs} + \text{gates} + \text{fanout branches} \\ &= 2 + 4 + 6 = 12\end{aligned}$$

Therefore,

$$\begin{aligned}\text{Number of single and multiple faults} &= 3^{\text{number of fault sites}} - 1 \\ &= 3^{12} - 1 = 531,440\end{aligned}$$

The circuit has 531,440 single and multiple stuck-at faults.



Circuit for Problem 4.5.

4.5 CMOS faults

(a) A two-input NAND gate is shown in the above figure. The following table gives tests for transistor stuck-open (sop) faults:

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

Notice that the sop faults of N1 and N2 have exactly the same tests. These two faults are equivalent. Equivalence of transistor faults is discussed in the following paper:
M.-L Flottes, C. Landrault and S. Provossoudovitch, "Fault Modeling and Fault Equivalence in CMOS Technology," *J. Electronic Testing: Theory and Applications*, vol. 2, pp. 229-241, August 1991.
(b) The following sequence of four vectors contains one vector pair for each fault in the above table:

11, 01, 11, 10

Notice that this sequence also detects all single stuck-at faults in the logic model of the NAND gate.
(c) A stuck-at fault in a signal affects two transistors in the two-input NAND gate. For example, the fault A s-a-1 will mean that N1 remains permanently shorted (N1-ssh) and P1 remains permanently open (P1-sop). The following table gives all equivalences:

Stuck-at fault	Equivalent transistor faults
A s-a-1	N1-ssh and P1-sop
B s-a-1	N1-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 and P2-sop

Notice that the three equivalent faults, A s-a-0, B s-a-0 and C s-a-0, are actually caused by different faulty transistors. They are detected by the same test (11).

4.6 Fault models

See Section 4.4 in the book.

4.7 Fault indistinguishability

Without loss of information we will write a function $f(V)$ as f. Thus, the left hand side of Equation 4.3 is:

$$
\begin{aligned}
&[f_0 \oplus f_1] \oplus [f_0 \oplus f_2] \\
&= [f_0\overline{f_1} + \overline{f_0}f_1] \oplus [f_0\overline{f_2} + \overline{f_0}f_2] \\
&= (f_0\overline{f_1} + \overline{f_0}f_1)\overline{(f_0\overline{f_2} + \overline{f_0}f_2)} + \overline{(f_0\overline{f_1} + \overline{f_0}f_1)}(f_0\overline{f_2} + \overline{f_0}f_2) \\
&= (f_0\overline{f_1} + \overline{f_0}f_1)\overline{(f_0\overline{f_2})}\,\overline{(\overline{f_0}f_2)} + \overline{(f_0\overline{f_1})}\,\overline{(\overline{f_0}f_1)}(f_0\overline{f_2} + \overline{f_0}f_2) \\
&= (f_0\overline{f_1} + \overline{f_0}f_1)(\overline{f_0} + f_2)(f_0 + \overline{f_2}) + (\overline{f_0} + f_1)(f_0 + \overline{f_1})(f_0\overline{f_2} + \overline{f_0}f_2) \\
&= (f_0\overline{f_1} + \overline{f_0}f_1)(\overline{f_0}\ \overline{f_2} + f_0f_2) + (\overline{f_0}\ \overline{f_1} + f_0f_1)(f_0\overline{f_2} + \overline{f_0}f_2) \\
&= f_0\overline{f_1}f_2 + \overline{f_0}f_1\overline{f_2} + \overline{f_0}\ \overline{f_1}f_2 + f_0f_1\overline{f_2} \\
&= (\overline{f_1}f_2)(f_0 + \overline{f_0}) + f_1\overline{f_2}(\overline{f_0} + f_0) \\
&= \overline{f_1}f_2 + f_1\overline{f_2} \\
&= f_1 \oplus f_2 \\
&= \text{Left hand side of Equation 4.4}
\end{aligned}
$$

This completes the derivation of Equation 4.4 from Equation 4.3.

4.8 Functional equivalence

Faulty functions for the circuit of Figure 4.12 corresponding to the two faults are:

$$
\begin{aligned}
i(c\ s-a-0) &= b(\overline{ab}) = \overline{a}b \\
i(f\ s-a-1) &= (a+b)\overline{a} = \overline{a}b
\end{aligned}
$$

The two faulty functions are indistinguishable and hence **the two faults are equivalent.**

4.9 Functional equivalence

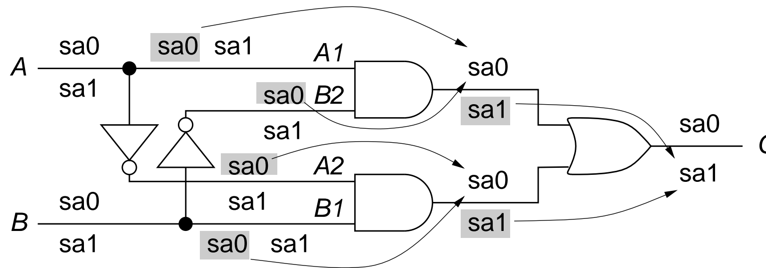
Faulty functions for the circuit of Figure 4.6 corresponding to the two faults are:

$$
\begin{aligned}
z(c\ s-a-1) &= \overline{\overline{\overline{ab}}.(\overline{\overline{ab}.b})} \\
&= \overline{ab.(ab+\overline{b})} = \overline{ab} \\
z(f\ s-a-1) &= \overline{ab}
\end{aligned}
$$

The two faulty functions are indistinguishable and hence **the faults are equivalent**.

4.10 Fault collapsing for test generation

The circuit of Figure 4.9 has 18 single stuck-at faults. Gate-level fault equivalence, as shown in the following figure, reduces the number to 12. The faults in shaded boxes have been collapsed as shown by arrows. Many ATPG and fault simulation



programs will collapse faults as shown above. However, functional fault collapsing can further reduce the number of faults to 10. As shown in Example 4.11 (see page 75 of the book), the s-a-1 faults on $A1$ and $B1$ are equivalent, and so are the s-a-1 faults on $A2$ and $B2$.

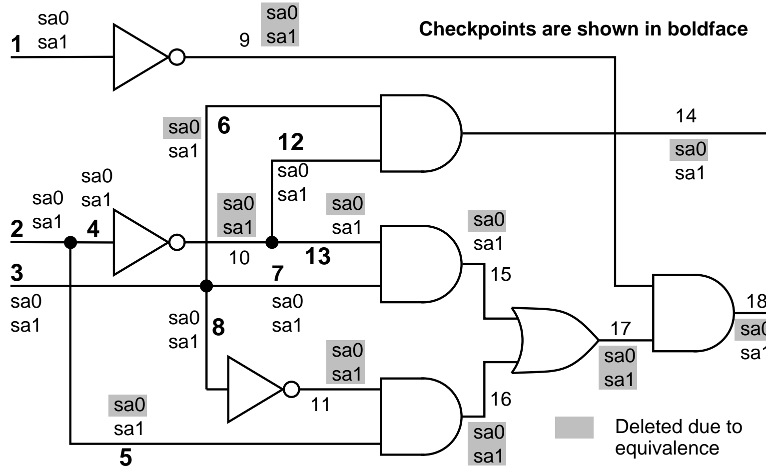
Whether we take the set of 12 faults or the set of 10 faults, their detection requires all four input vectors.

4.11 Equivalence and dominance fault collapsing

(a) The given circuit is shown below with fault sites marked by numbers. The number of potential fault sites is 18. The total number of faults is 36.

(b) The figure shows deletion of equivalent faults using an output to input pass. Of the 36 faults, 20 remain, giving a collapse ratio $20/36 = 0.56$.

(c) Checkpoint lines are shown by boldface numbers. These are three PIs and seven fanout branches. Line 2 fans out to 4 and 5. Line 3 fans out to 6, 7 and 8. Line 10 fans out to 12 and 13. There are ten checkpoints and 20 checkpoint faults. Further, s-a-0 faults on lines 6 and 12 are equivalent and any one of them can be chosen. Similarly, s-a-0 faults on 7 and 13 are equivalent, and so are s-a-0 on 5 and s-a-1 on 8. Thus, the size of the fault set is reduced to 17, giving a collapse ratio $17/36 = 0.47$.



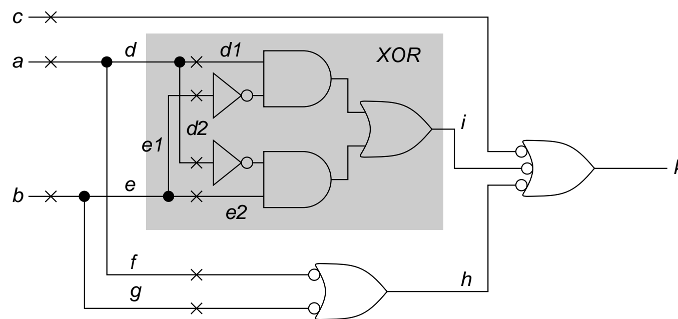
Circuit for Problem 4.11: (b) Equivalence collapse ratio = 20/36 = 0.56
(c) Dominance (uncollapsed faults at checkpoints) collapse ratio = 17/36 = 0.47

4.12 Dominance fault collapsing

(a) Checkpoints are defined for the signals in a combinational circuit. These signals are the interconnects between Boolean gates, a fact not always explicitly stated. To avoid ambiguity, the definition on page 78 of the book should read as:

Definition 4.7 *Checkpoints. Primary inputs and fanout branches of a combinational circuit consisting only of Boolean gates are called the* checkpoints.

To find checkpoints of the circuit of Figure 4.12, we must replace the exclusive-OR (XOR) function by a primitive Boolean gate implementation. AND, OR, NAND, NOR and NOT are called the *primitive* Boolean gates. Functions such as XOR are sometimes referred to as *complex* gates. In the following figure, we have assumed one such implementation. Our result is, therefore, based on this assumption. Other implementations of the XOR function are possible and can give a different set of checkpoints.



There are nine checkpoints in this circuit. These include three primary inputs, a, b and c, and six fanout branches, $d1$, $d2$, f, $e1$, $e2$ and g. The checkpoint fault set consists of eighteen faults – s-a-0 and s-a-1 faults on the nine lines.

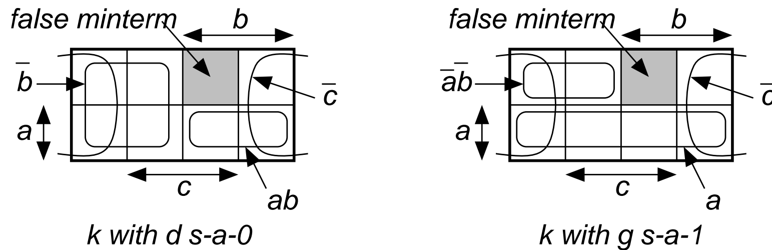
Notice that lines d and e of the original circuit are not checkpoints. If we did

not model the XOR block with Boolean gates, then those lines will appear to be checkpoints, whose number will be fourteen. However, detection of those faults will not guarantee detection of faults on the fanouts that are internal to the XOR block. Considering the Boolean gate structure, a fault on d corresponds to a simultaneous (multiple) fault on $d1$ and $d2$ and, in general, the detection of a multiple fault is not equivalent to detection of the component faults.

(b) We evaluate the output function k corresponding to the two faults:

$$\begin{aligned} k(d\ s-a-0) &= \bar{c} + \bar{b} + \overline{\bar{a} + \bar{b}} \\ &= \bar{c} + \bar{b} + ab \\ k(g\ s-a-1) &= \bar{c} + \overline{a\bar{b} + \bar{a}b} + a \\ &= \bar{c} + \bar{a}\bar{b} + a \end{aligned}$$

The two faulty functions are shown by Karnaugh maps below. In both cases, the functions have exactly one false minterm, $\bar{a}bc$. **Since the two faulty functions are identical the corresponding faults are equivalent**.

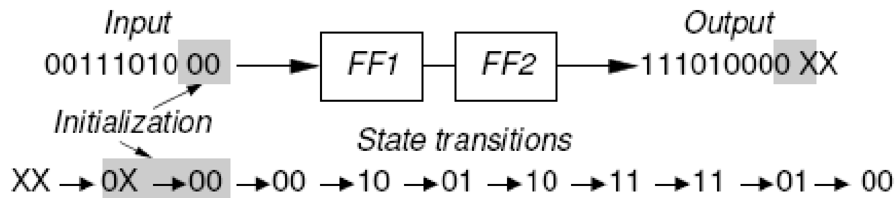


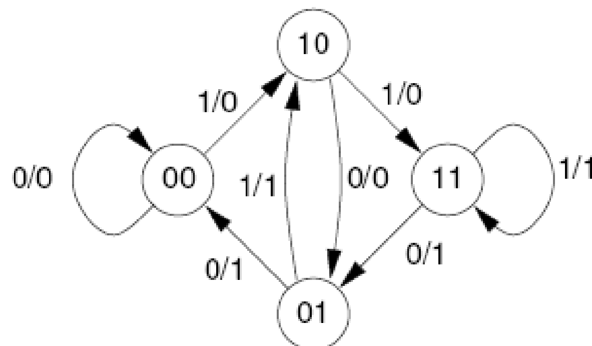
k with d s-a-0   *k with g s-a-1*

Note: this type of fault equivalence is functional and is often difficult to find by typical fault analysis tools, which rely on structurally identifiable equivalences.

Problem 5.2

The following figure shows a two-bit shift register. Initially, both flip-flops are in the 0 state. The first two 0 inputs initialize the flip-flops to the 00 state. Subsequent inputs, outputs and state transitions are shown in the figure.





State diagram of 2–bit shift register.

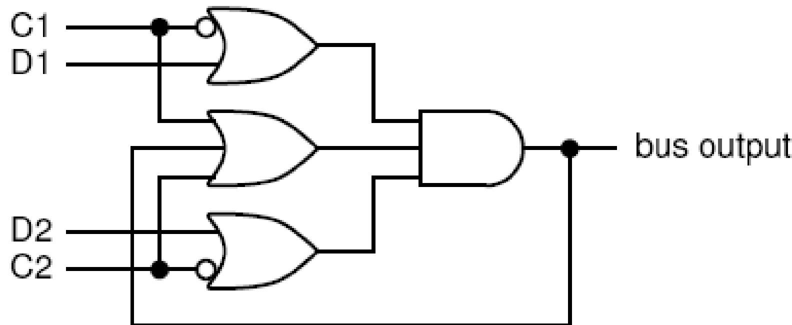
Problem 5.4

The longest path in the circuit (see Figure 5.2) is C_0 to C_4. The delay of this path should be tested for both rising and falling transitions. As shown in Example 5.3, the path delay for a rising transition is tested by vector 2 followed by 6, which causes the transition to ripple through the path. Similarly, the path delay for a falling transition can be tested by vector-pair, 6 followed by 2. From Table 5.2, vectors 1 through 6 cover all stuck-at faults. Since the circuit is combinational, these vectors can be applied in any order. We construct a sequence of seven vectors using these six vectors that contains the two delay test vector-pairs. The sequence is 1, 2, 6, 2, 3, 4, 5.

Note: If we use the result of Table 5.3, a sequence of six vectors for all stuck-at faults and two path delay faults can be constructed.

Problem 5.8

The following schematic shows a logic model for a bus with memory. When both drivers feed data to the bus, i.e., $C1 = C2 = 1$, $D1 \cdot D2$ appears at the output, assuming a 0-dominance. When both drivers are turned off, i.e., $C1 = C2 = 0$, the output retains its value through feedback. When only one driver is on, the corresponding data input appears at the output.



This model represents most of the characteristics of a MOS bus, with the exception of bidirectionality. One problem with it is that it is an asynchronous sequential circuit and cannot be correctly simulated by some simulators. An event-driven logic simulator can simulate it, but will be inefficient in comparison with synchronous circuit simulation.

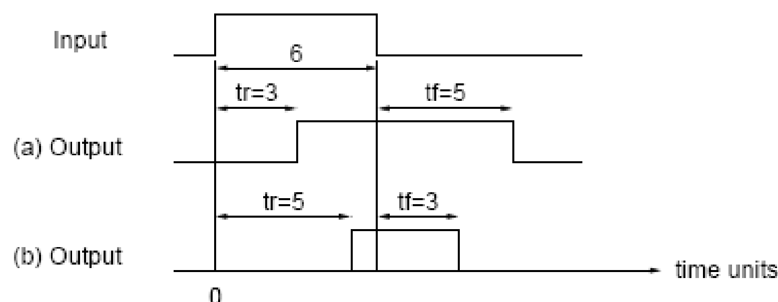
Problem 5.10

With the given inputs, 00, and output X, when the clock is applied the circuit will not be initialized. The reason is that in a three-state logic system the inversion of X is also X.

The circuit can be initialized to a 1 output by clocking the flip-flop when a 11 input is applied. Then, if we change the input to 10 and clock the flip-flop, the output will become 0. These two vectors can be correctly simulated by a three-state logic simulator.

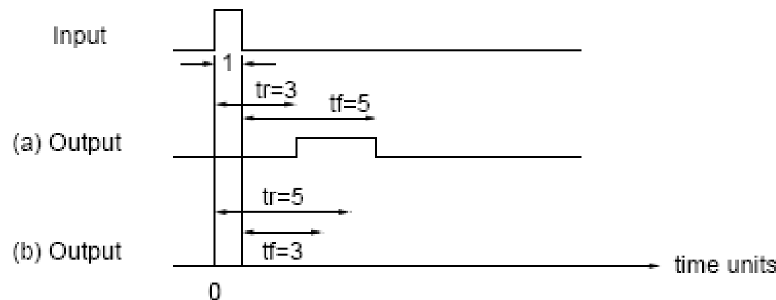
5.11 (10 points)

The two cases are sketched below. The rise and fall delays of the OR gate are denoted by tr and tf, respectively. In (a) the output pulse width is 8 units and in (b) it is 4 units.



5.12 (15 points)

The two cases are sketched below. The rise and fall delays of the OR gate are denoted by tr and tf, respectively. In (a), a rise is first scheduled to occur at 3 time units after the rising edge of the input. Before this rise takes place, the input falls at 1 unit, and reschedules a falling output at time 6 units. A conservative simulator produces an unknown (X) output between 3 and 6 units of time. This is shown as a level between logic 0 and 1 in the following figure.



In (b), the output cannot rise until 5 units of time. Meanwhile, at time unit 1, a fall is scheduled to be completed at time unit 4. Thus, the output does not change at all.

Case (b) is an example of a pulse being filtered by a slow gate. In simulators, this phenomenon is referred to as *spike suppression.* The actual waveform produced by the simulator depends upon the specific assumptions made. In pessimistic simulation, a pulse of ambiguous

height may be produced as in case (a) above. In optimistic simulation, the output may remain unchanged if the input pulse width is smaller than the gate delay.

Problem 5.16

Since no fault dropping is used, the serial fault simulator must simulate the entire circuit $n+1$ times. Assuming the CPU time for one simulation with all vectors is t, total time of serial fault simulation is given by,

$$T(serial) = t(n+1)$$

Using CPU time t, the parallel simulator processes $w-1$ faults. Thus, it will make $n/(w-1)$ such passes, requiring total time,

$$T(parallel) = \frac{tn}{w-1}$$

Therefore,

$$\frac{T(serial)}{T(parallel)} = \frac{(n+1)(w-1)}{n}$$

5.17 (15 points)

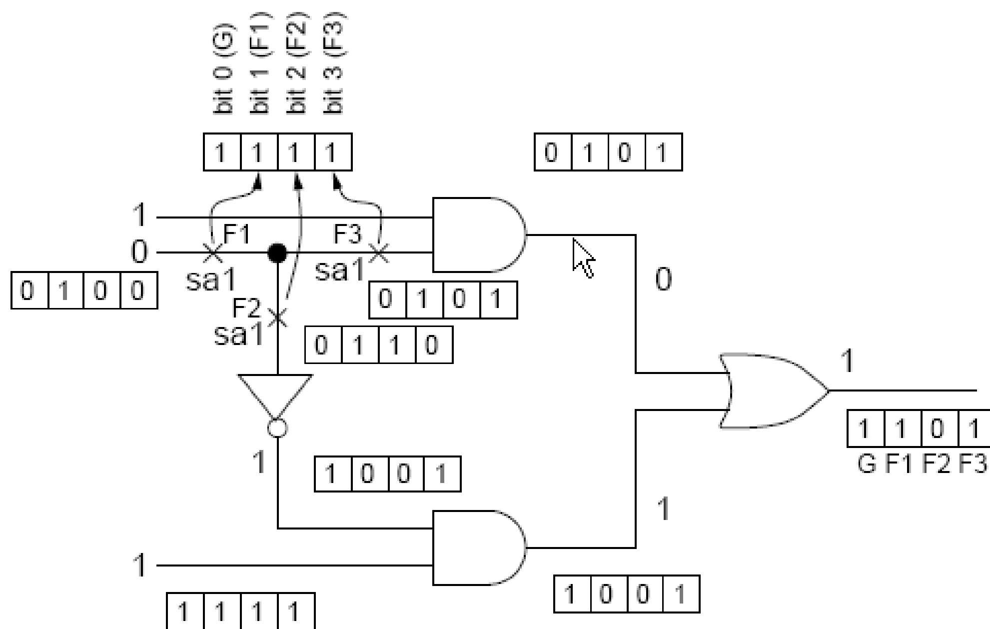
The circuit of Figure 5.22 is shown below. The bits of the four-bit word are assigned as follows:

Bit 0: G, good circuit

Bit 1: Faulty circuit with fault F1, second input s-a-1

Bit 2: Faulty circuit with fault F2, input to inverter s-a-1

Bit 3: Faulty circuit with fault F3, second input of first AND gate s-a-1



The figure shows the good and faulty circuit values for each signal by a four-bit word. A comparison among bits of the word at the primary output indicates that only the bit corresponding to F2 differs with the good circuit output. Hence, the vector 101 detects F2 but does not detect F1 and F3.

Problem 5.25 Fault sampling

Since the size of fault population ($N_p = 10^5$) is very large compared to the sample size ($N_s = 4,000$), we use the approximation of Equation 5.5 (page 123 in the book.)

$$\text{Sample coverage, } x = \frac{3,900}{4,000} = 0.975$$

Using Equation 5.8 (see page 123 in the book), we get

$$\begin{aligned}
3\sigma \text{ coverage estimate} &= x \pm \frac{4.5}{N_s}\sqrt{1 + 0.44 N_s x(1-x)} \\
&= 0.975 \pm \frac{4.5}{4,000}\sqrt{1 + 0.44 \times 4,000 \times 0.975 \times 0.025} \\
&= 0.975 \pm 0.0075 \text{ or } 97.50 \pm 0.75 \text{ percent}
\end{aligned}$$

Problem 5.26 Fault sampling

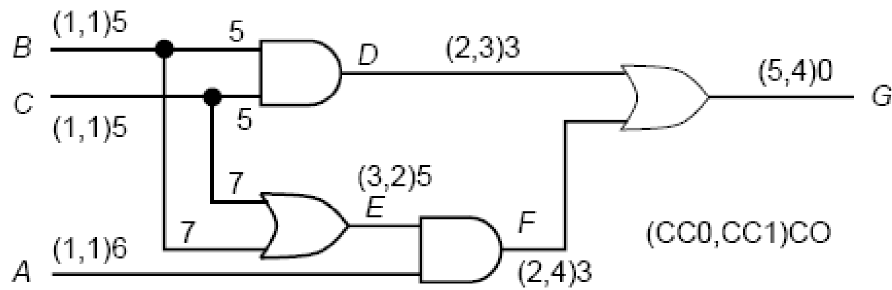
Assuming that the fault sample size is much smaller than the total fault population, i.e., $N_s \ll N_p$, we use the result of Equation 5.9 (page 124 in the book), which can be written as,

$$\text{Sample size, } N_s = \frac{4.5^2}{\Delta^2} 0.44 x(1-x)$$

where $\pm\Delta$ is the 3σ range of the coverage estimate and x is the sample coverage. Using the given data, $\Delta = 0.02$ and $x = 0.70$, we obtain

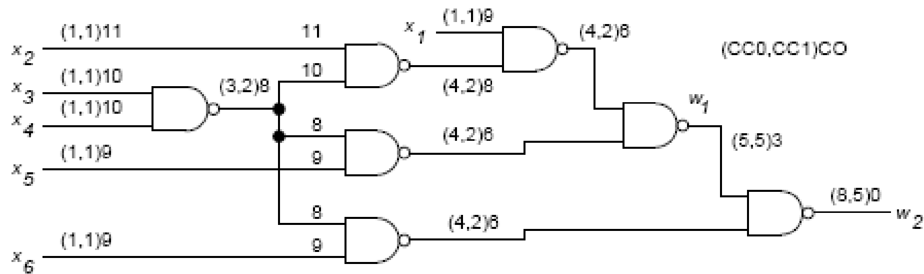
$$N_s = \frac{4.5^2 \times 0.44 \times 0.7 \times 0.3}{0.02^2} = 4,678 \text{ faults}$$

Problem 6.2 SCOAP



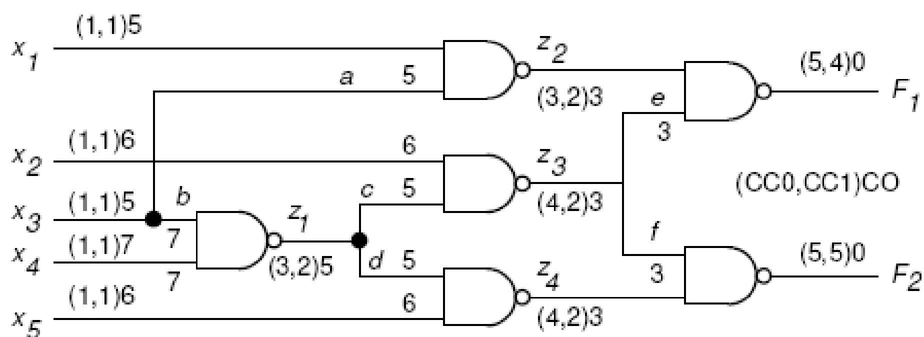
Circuit of Figure 6.20 with combinational SCOAP measures.

(Bushnell and Agrawal) Problem 6.3



Circuit of Figure 6.21 with combinational SCOAP measures.

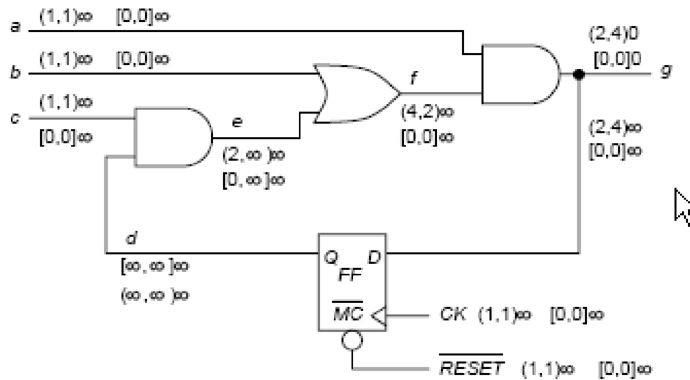
Problem 6.4 SCOAP



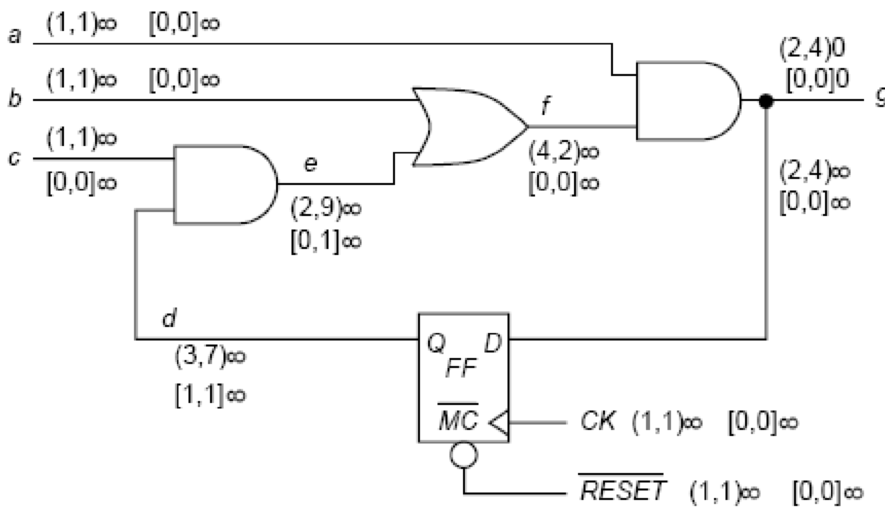
Circuit of Figure 6.22 with combinational SCOAP measures.

Problem 6.7 SCOAP

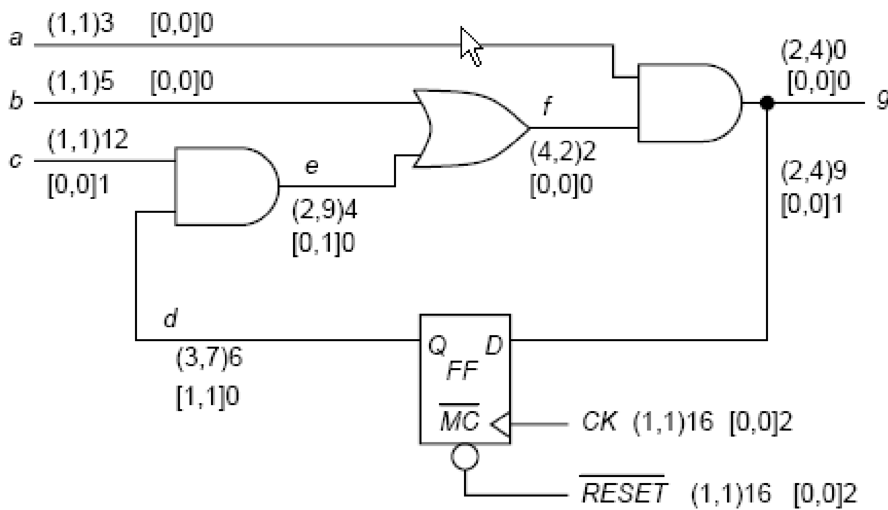
The steps of calculation for SCOAP testability measures are shown in the three figures that follow. Combinational measures are shown as $(CC0, CC1)CO$ and sequential measures as $[SC0, SC1]SO$.



Circuit of Figure 6.25: PI and PO initialization and first controllability pass.



Circuit of Figure 6.25: Converged controllability values.



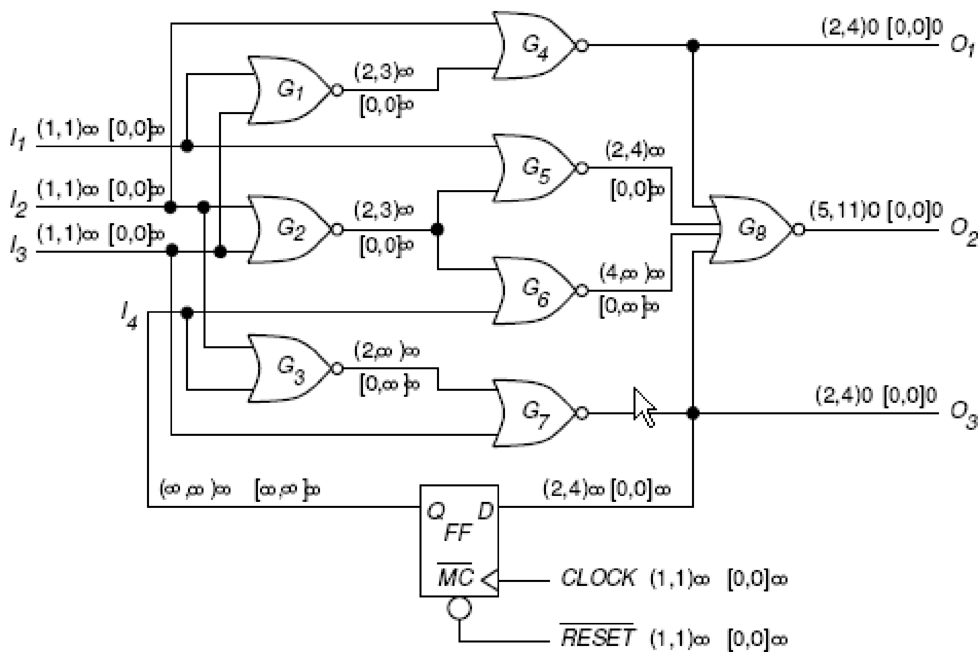
Circuit of Figure 6.25: All controllability and observability values.

(Bushnell and Agrawal) Problem 6.9

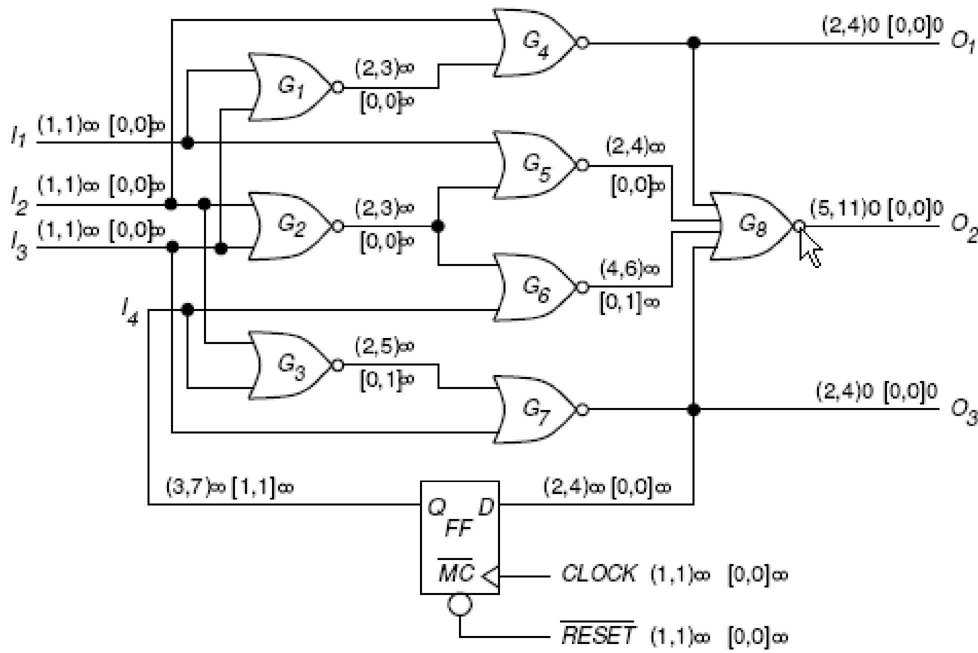
The steps of calculation for SCOAP testability measures are shown in the three figures that follow. Combinational measures are shown as $(CC0, CC1)CO$ and sequential measures as $[SC0, SC1]SO$.

Problem 6.8 SCOAP

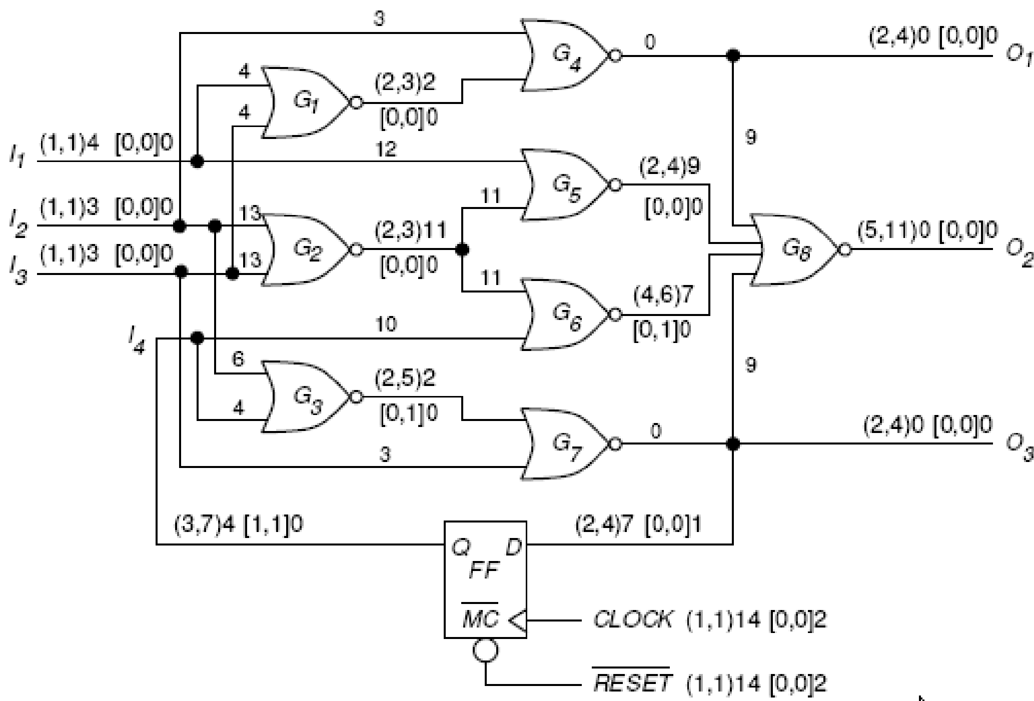
The steps of calculation for SCOAP testability measures are shown in the three figures that follow. Combinational measures are shown as $(CC0, CC1)CO$ and sequential measures as $[SC0, SC1]SO$.



Circuit of Figure 6.26: PI and PO initialization and first controllability pass.



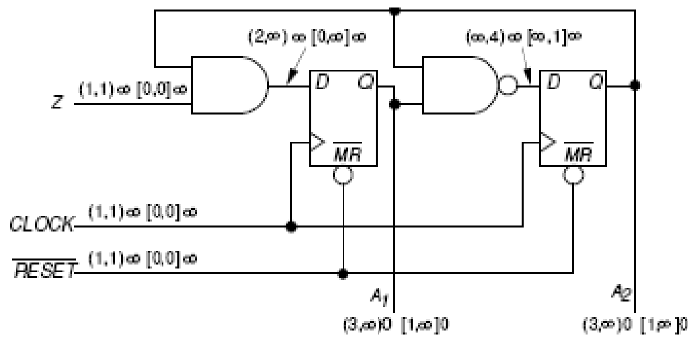
Circuit of Figure 6.26: Converged controllability values.



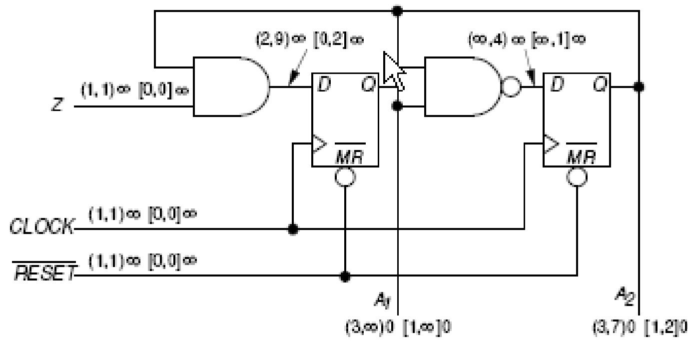
Circuit of Figure 6.26: All controllability and observability values.

Problem 6.13 SCOAP

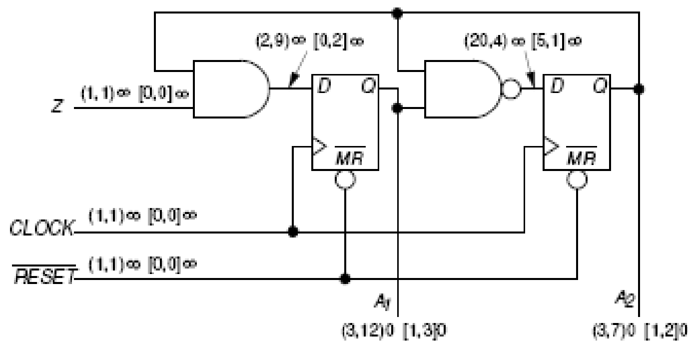
The steps of calculation for SCOAP testability measures are shown in the four figures that follow. Combinational measures are shown as $(CC0, CC1)CO$ and sequential measures as $[SC0, SC1]SO$.



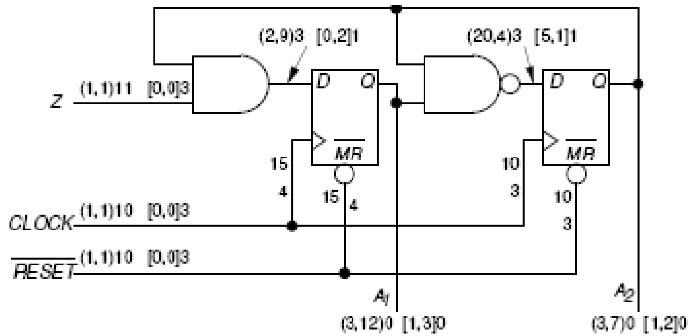
Circuit of Figure 6.31: Initialization and first controllability pass.



Circuit of Figure 6.31: Continuation of controllability calculation.



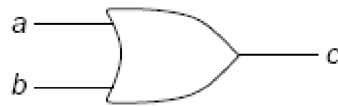
Circuit of Figure 6.31: Stabilized controllability values.



Circuit of Figure 6.31: All controllability and observability values.

Problem 7.2 Stuck-at fault testing

(a) *Three tests for a two-input OR gate:*



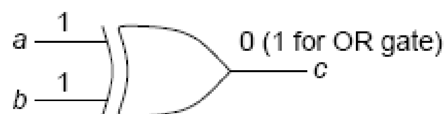
Vector number	a	b	c	Collapsed faults tested
1	0	0	$\overline{D}$	a sa1, b sa1, c sa1
2	0	1	D	b sa0, c sa0
3	1	0	D	a sa0, c sa0

(b) *Gate replacements:*

OR replaced by:	Test results		
	Vector 1	Vector 2	Vector 3
AND	pass	fail	fail
NAND	fail	pass	pass
NOR	fail	fail	fail

The three-vector test will detect the error if the OR gate were to be replaced by an AND, NAND or NOR gate.

(c) *OR gate replaced by an exclusive-OR gate:* All three vectors will produce the same output as that of the OR gate. Therefore, this error will not be detected. It is necessary to include a fourth vector 11 to detect this error. The addition of the



fourth vector makes the vector set exhaustive, which completely verifies the truth table of the gate.

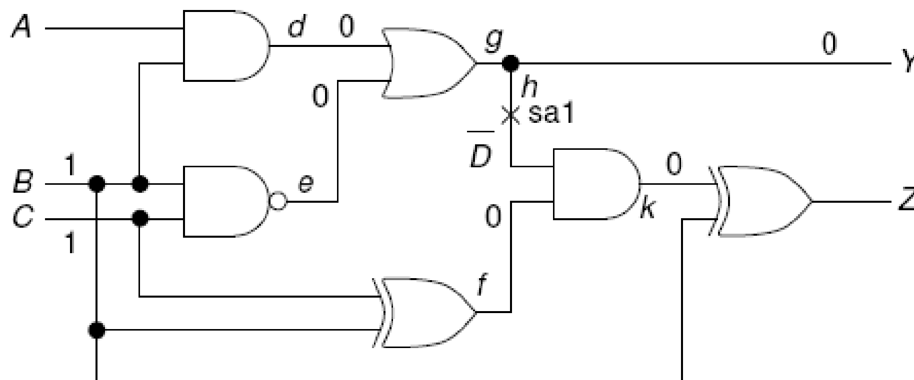
Note: In a simulation-based comparison of two circuits to establish logic equivalence, a good (*though not complete*) heuristic is to use a vector set that covers all single stuck-at faults in both circuits. See the paper: V. D. Agrawal, "Choice of Tests for Logic Verification and Equivalence Checking and the Use of Fault Simulation," *Proc. 13th Int. Conf. VLSI Design*, 2000, pp. 306-311.

Problem 7.3 D-ALG

We level order the signals and proceed as follows:

Step no.	Action	A	B	C	d	e	f	g	Y	h	k	Z	D front.	Impl. stack
1	Fault Activation							0	0	$\overline{D}$			k	$g=0$
	Immediate impl.				0	0		0	0	$\overline{D}$			k	$g=0$
	Immediate impl.		1	1	0	0		0	0	$\overline{D}$			k	$g=0$
	Immediate impl.		1	1	0	0	0	0	0	$\overline{D}$			k	$g=0$
	Immediate impl.		1	1	0	0	0	0	0	$\overline{D}$	0		ϕ	$g=0$
	Immediate impl.		1	1	0	0	0	0	0	$\overline{D}$	0	1	ϕ	$g=0$

The fault is redundant, because the D-frontier disappeared. No backtracks. Signals are shown in the following figure.

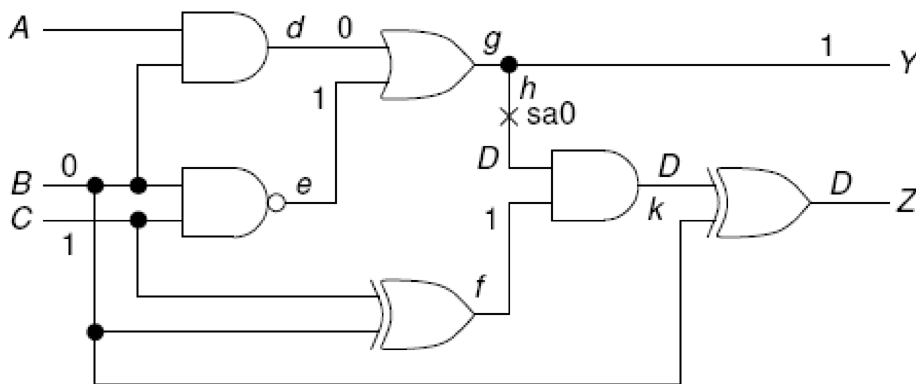


Problem 7.4 D-ALG

We level order the signals and proceed as follows:

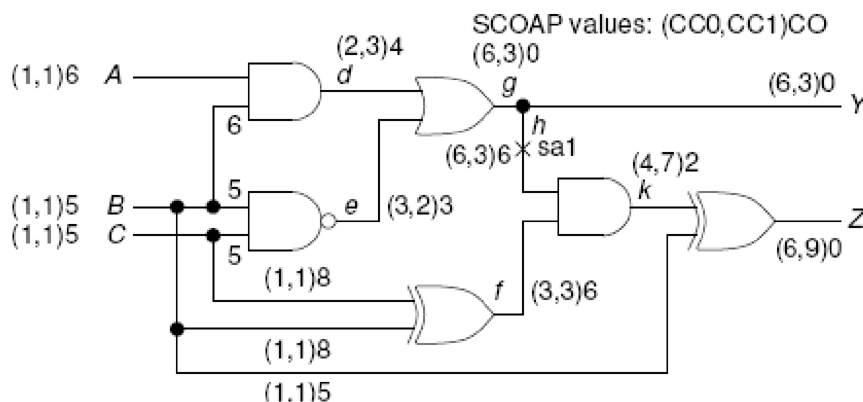
Step no.	Action	A	B	C	d	e	f	g	Y	h	k	Z	D front.	Impl. stack
1	Fault activation							1	1	D			k	$g=1$
2	D-drive $h \rightarrow k$						1	1	1	D	D		Z	$f=1$ $g=1$
3	D-drive $k \rightarrow Z$		0				1	1	1	D	D	D	PO	$B=0$ $f=1$ $g=1$
	Immediate Impl.		0		0		1	1	1	D	D	D	PO	"
	Immediate Impl.		0		0	1	1	1	1	D	D	D	PO	"
	Immediate impl.		0	1	0	1	1	1	1	D	D	D	PO	"

The test is: $A = X$, $B = 0$, $C = 1$ as shown in the following figure; 0 backtracks.



Problem 7.5 PODEM

The figure below shows the SCOAP testability measures used for guiding PODEM.



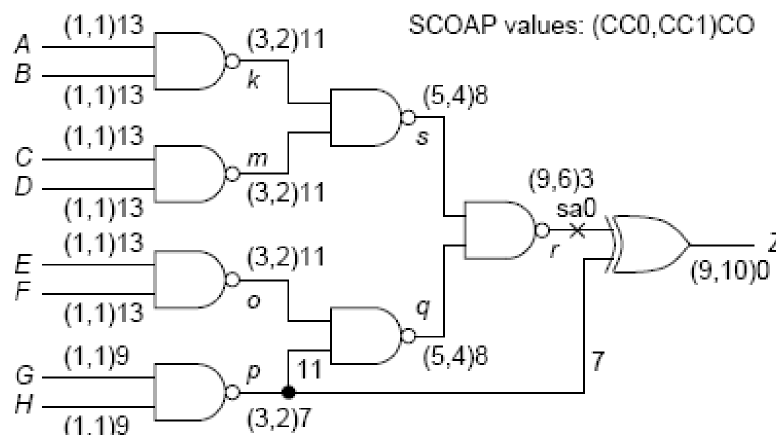
The steps of the PODEM algorithm are recorded in the following table:

Step No.	Objec-tive	Action	Imp. stack	A	B	C	d	e	f	g	h	k	Y	Z	D front.	X path
				Implied signal values												
1	$g = 0$	Backtrace	$B = 1$		1										ϕ	ok
2	$g = 0$	Backtrace	$C = 1$ $B = 1$		1	1		0	0			0		1	ϕ	none
3	$g = 0$	Backtrack	$C = 0$ $B = 1$		1	0		1	1	1	1	1	1	0	ϕ	none
4	$g = 0$	Backtrack	$B = 0$		0		0	1		1	1		1		ϕ	none
5	$g = 0$	Backtrack	Empty													
Algorithm termination: Objective g=0 is impossible; fault h s-a-1 is redundant.																

Explanation: An *X-path* is a path from the fault site to a PO, such that the signals on it are either faulty states (D or $\overline{D}$) or *undetermined*. An "ok" for *X-path* in the table means that one or more such paths exist. Having no X-path is a reason for *backup* because its existence is a necessary condition for the detection of the fault. When a series of backups leads to an *empty* stack, it indicates that the objective $g = 0$ is impossible. As a result, **the fault h s-a-1 cannot be activated and, hence, it is redundant.** Three backtracks.

Problem 7.6 PODEM

The figure below shows the SCOAP testability measures used for guiding PODEM.



The steps of the PODEM algorithm are recorded in the following table:

Step No.	Objec-tive	Action	Imp. stack	Implied signal values $ABCDEFGHkmopqsrZ$	D front.	X path
1	$r = 1$	Backtrace	$E = 0$	$E = 0, o = 1$	ϕ	ok
2	$r = 1$	Backtrace	$G = 0$ $E = 0$	$E = 0, G = 0, o = 1, p = 1$ $q = 0, r = 1, Z = \overline{D}$	PO	ok
Algorithm termination: Fault detected with 0 backtracks. *Test is* $\{ABCDEFGH\} = \{XXXX0X0X\}$						

Explanation: An *X-path* is a path from the fault site to a PO, such that the signals on it are either faulty states (D or $\overline{D}$) or *undetermined*. An "ok" for *X-path* in the table means that one or more such paths exist. Having no X-path is a reason for *backup* because its existence is a necessary condition for the detection of the fault.

Dominators: Problem 7.7b

For each gate in Figure 7.41, identify its absolute dominators.

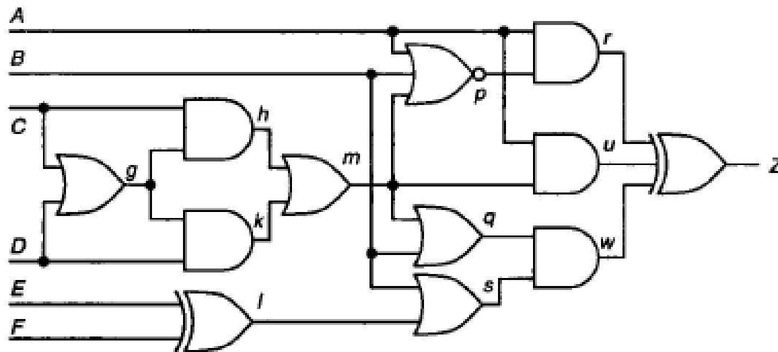


Figure 7.41: Circuit for Problems 7.7 through 7.12.

A: Gate Z is an absolute dominator.
B: Gate Z is an absolute dominator.
C: Gates m and Z are absolute dominators.
D: Gates m and Z are absolute dominators.
E: Gates l, s, w and Z are absolute dominators.
F: Gates l, s, w and Z are absolute dominators.
g: Gates m and Z are absolute dominators.
h: Gates m and Z are absolute dominators.
k: Gates m and Z are absolute dominators.
l: Gates s, w and Z are absolute dominators.
m: Gate Z is an absolute dominator.

p: Gates r and Z are absolute dominators.
q: Gates w and Z are absolute dominators.
s: Gates w and Z are absolute dominators.
r: Gate Z is an absolute dominator.
u: Gate Z is an absolute dominator.
w: Gate Z is an absolute dominator.
z: none.

(Bushnell and Agrawal) Problem 7.8
Note : While you are performing the PODEM algorithm, follow the rules given below.

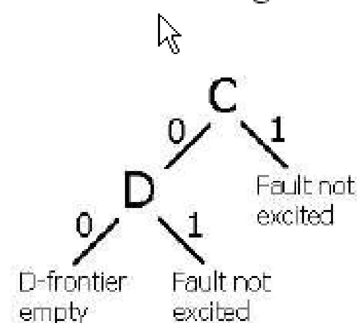
```
- Order: Try to excite the fault first then propagate.
- Backtrace: Follow a path from the objective to a primary input while
  always following the alphabetical order (e.g. if a gate has input A and B,
  backtrace on that gate goes to line A first.
- PI assignment: Always assign 0 first, then assign 1 in case of backtrack.
- Choice of D or D_bar : Always try to propagate a D or D_bar from the
  D-frontier which has the shortest path to the primary output.
  In the case of tie, follow the alphabetical order.
```

The following table gives the steps of PODEM (see Problem 7.5 for an explanation of *X-path*, it's not required):

Step No.	Objective	Action	Imp. stack	Implied signal values $ABCDEFghklmpqsruwZ$	D front.	X path
1	$g = 0(\overline{D})$	Backtrace	$C = 0$	$C = 0, h = 0$	ϕ	ok
2	$g = 0(\overline{D})$	Backtrace	$D = 0$ $C = 0$	$C = 0, D = 0, g = 0(\overline{D})$ $h = 0, k = 0, m = 0, u = 0$	ϕ	none
3	$g = 0(\overline{D})$	Backtrack	$D = 1$ $C = 0$	$C = 0, D = 1, g = 1, h = 0$ $k = 1, m = 1, p = 0, q = 1, r = 0$	ϕ	none
4	$g = 0(\overline{D})$	Backtrack	$C = 1$	$C = 1, g = 1, h = 1, m = 1$ $p = 0, q = 1, r = 0$	ϕ	none
5	$g = 0(\overline{D})$	Backtrack	Empty			
Algorithm termination: $g = 0(\overline{D})$ with X-path impossible; fault g s-a-1 is redundant. 3 backtracks.						

PODEM: Problem 7.8. Do not use any internal node assignments or mandatory value assignments. (Make an arbitrary decision when a choice is available during back-trace). Generate a test with the PODEM ATPG algorithm for the fault *g* s-a-1 in Figure 7.41.

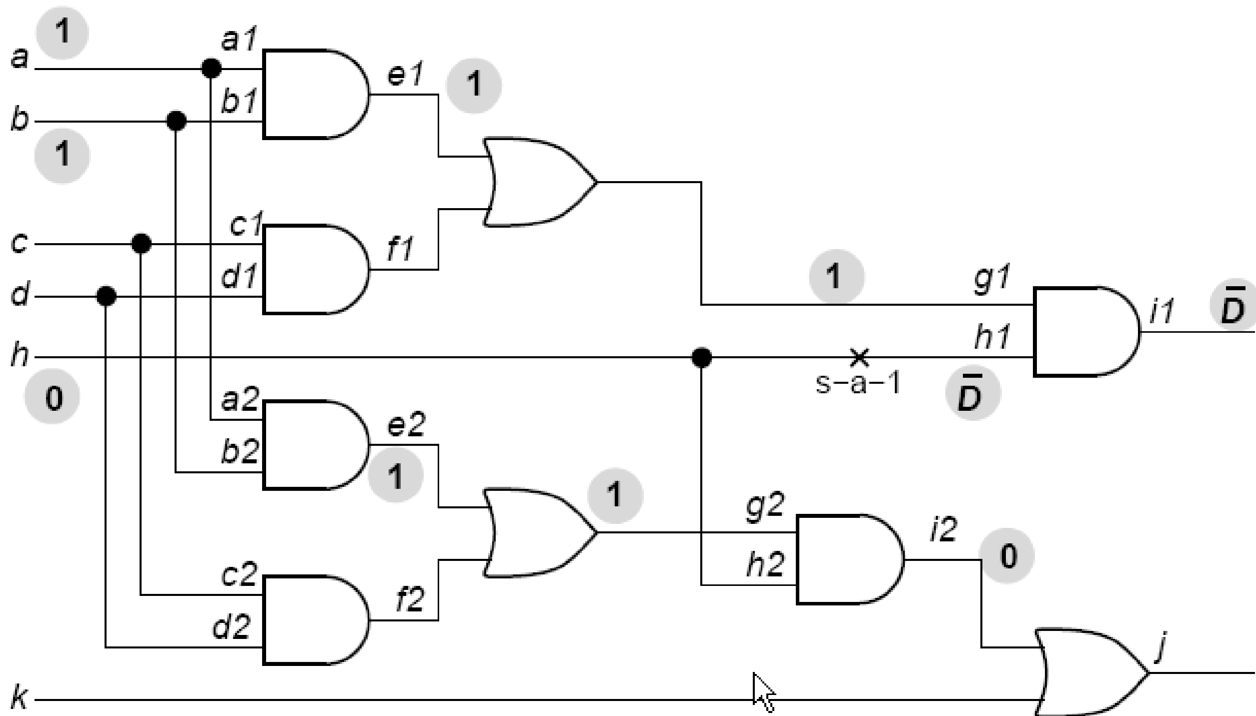
See problem 2) for circuit diagram.
Select path: g → h → m → u → Z
Initial objective: g → 0 (excite s-a-1 fault)
C → 0 (path through h is blocked)
D → 0 (paths through h, k are blocked, D-frontier empty)
D → 1 (fault not excited)
C → 1 (fault not excited)
PODEM stops (fault untestable)



(Bushnell and Agrawal) Problem 7.13

Step	Action	Impl. stack	Forward implications	D-frontier
1	Fault act.	$h = 0$	$h = 0, h1 = \overline{D}, i2 = 0$	$i1$
2	D-prop.	$g1 = 1, h = 0$	$g1 = 1, h = 0, h1 = \overline{D}$ $i1 = \overline{D}, i2 = 0$	PO
3	Justify	$e1 = 1, g1 = 1$ $h = 0$	$e1 = 1, g1 = 1, h = 0$ $h1 = \overline{D}, i1 = \overline{D}, i2 = 0$	PO
4	Justify	$a = 1, b = 1$ $e1 = 1, g1 = 1$ $h = 0$	$a = 1, b = 1, e1 = 1, g1 = 1$ $e2 = 1, g1 = 1, g2 = 1$ $h = 0, h1 = \overline{D}, i1 = \overline{D}$ $i2 = 0$	PO
Test found: $(a, b, c, d, h, k) = (1, 1, X, X, 0, X)$; $i1 = \overline{D}$				

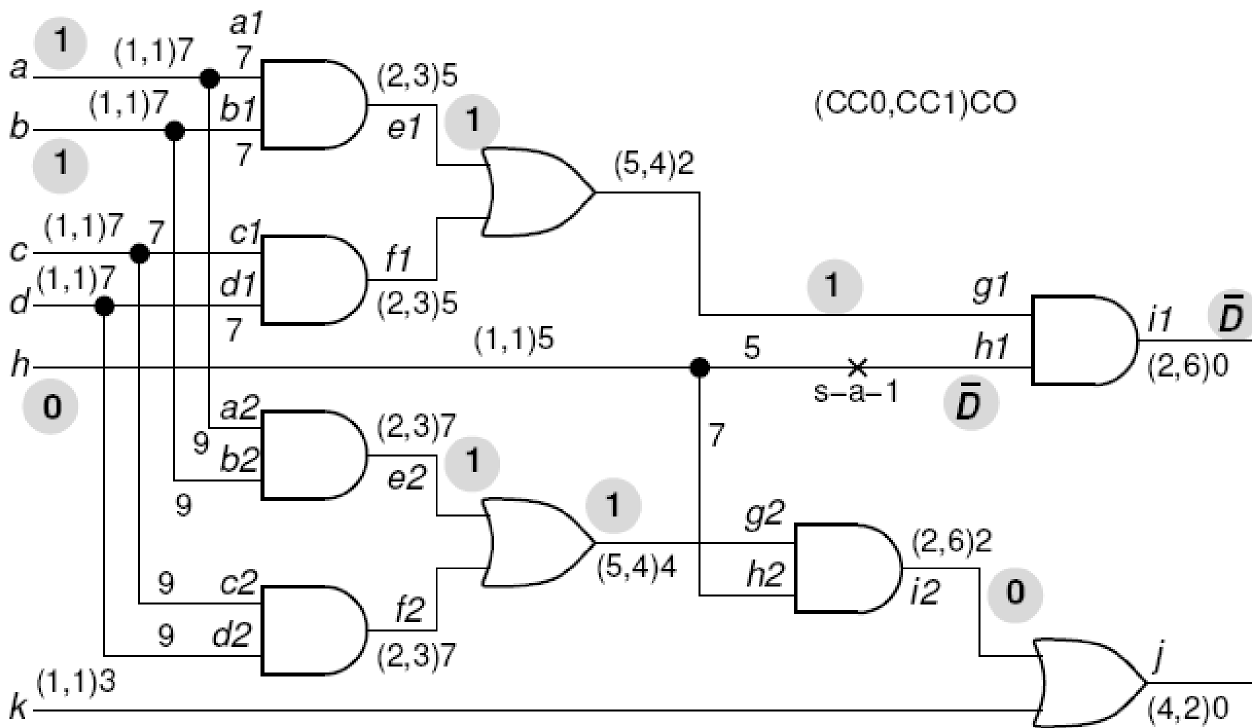
The above figure shows the circuit and the signal values specified by D-algorithm.



Problem 7.14 PODEM

Step	Objective (goal)	Impl. stack	Forward implications	D frontier	X path
1	Fault act.	$h=0$	$h=0, h1=\overline{D}, i2=0$	$i1$	ok
2	Fault prop. $g1=1$	$h=0, a=1$	$a=1, h=0, h1=\overline{D}$ $i2=0$	$i1$	ok
3	Fault prop. $g1=1$	$h=0, a=1$ $b=1$	$a=1, b=1, e1=1$ $e2=1, g1=1, g2=1$ $h=0, h1=\overline{D}, i1=\overline{D}$ $i2=0$	PO	ok
Test found: $(a,b,c,d,h,k)=(1,1,X,X,0,X)$; $i1=\overline{D}$					

The following figure shows the SCOAP testability measures used to guide the PODEM algorithm, and the signal values detremined.



This test is found without any backtracks.

7.17 SOCRATES (15 points)

To obtain a test for the fault n s-a-1 in the circuit of Figure 7.24 (see page 190 of the book and the figure below), we perform static learning:

Signal	Learned implications	Signal	Learned implications
$B = 0$	$(m = 0) \Rightarrow (B = 1)$ $(q = 1) \Rightarrow (B = 1)$ $(r = 1) \Rightarrow (B = 1)$ $(s = 0) \Rightarrow (B = 1)$ $(v = 1) \Rightarrow (B = 1)$	$X = 0$	$(r = 1) \Rightarrow (X = 1)$ $(Y = 1) \Rightarrow (X = 1)$ $(v = 1) \Rightarrow (X = 1)$ $(q = 1) \Rightarrow (X = 1)$ $(s = 1) \Rightarrow (X = 1)$ $(m = 0) \Rightarrow (X = 1)$
$d = 1$	$(m = 0) \Rightarrow (d = 0)$ $(q = 1) \Rightarrow (d = 0)$ $(r = 0) \Rightarrow (d = 0)$ $(s = 1) \Rightarrow (d = 0)$ $(X = 1) \Rightarrow (d = 0)$ $(Y = 1) \Rightarrow (d = 0)$ $(v = 1) \Rightarrow (d = 0)$ $(Z = 0) \Rightarrow (d = 0)$	$Y = 0$	$(X = 1) \Rightarrow (Y = 1)$ $(v = 1) \Rightarrow (Y = 1)$ $(r = 0) \Rightarrow (Y = 1)$ $(d = 0) \Rightarrow (Y = 1)$ $(m = 0) \Rightarrow (Y = 1)$
$Z = 1$	$(q = 1) \Rightarrow (Z = 1)$		

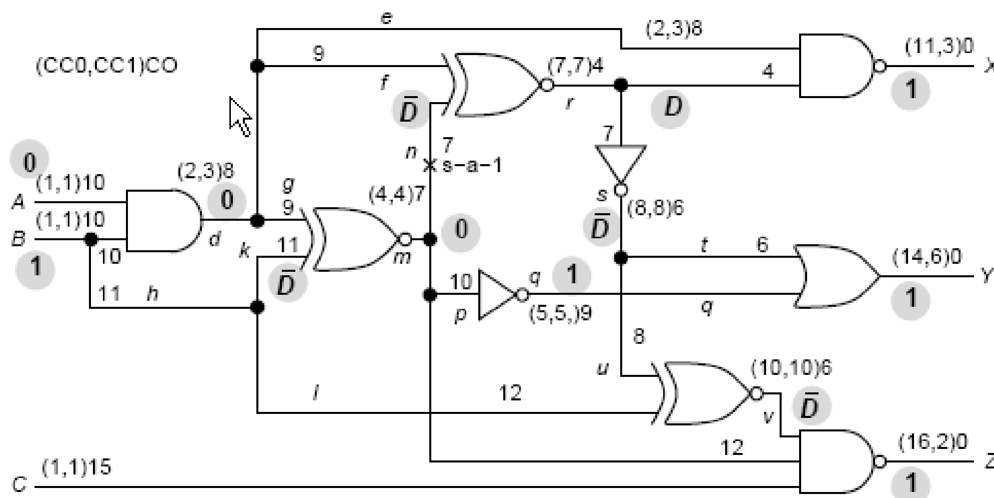
Step 1: Goal – sensitize fault, $m = 0$

Static learning – $B = 1$

Implications – $d = 0, X = 1, Y = 1, A = 0, r = D, q = 1,\ m = 0, s = \overline{D},\ v = \overline{D}, Z = 1$

D-frontier – ϕ (null)

Redundant fault, because D-frontier vanishes at gate Z, no decision alternatives. No need for dynamic learning, no use of the constructive dilemma or Modus Tollens. No backtracks.

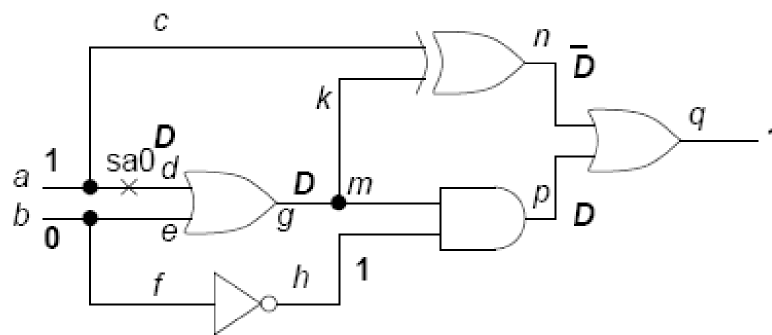


7.19 Redundancy proofs (20 points)

(1) Proof of d s-a-0 redundant using PODEM.

Step 1: Goal – sensitize fault. Objective – $d = 1$. Backtrace – Implication stack – $a = 1$. Implication – $d = D$. D-frontier – g.

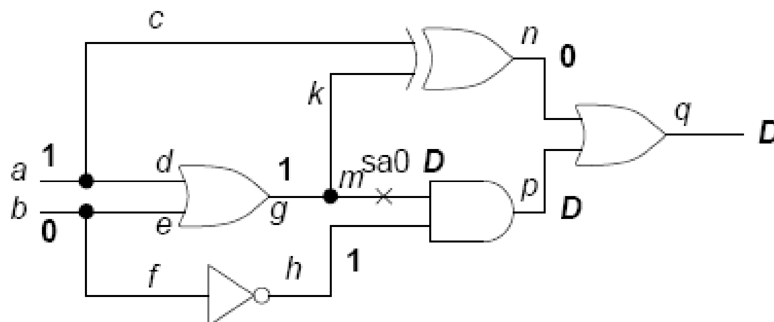
Step 2: Goal – propagate fault. Objective – $e = 0$. Backtrace – Implication stack – $a = 1, b = 0$.
Implications – $d = D, g = D, h = 1, n = \overline{D}, p = D, q = 1$.
D-frontier – ϕ.
Fault proved redundant because D-frontier disappears at q – no alternative assignments possible.



(2) Proof of m s-a-0 testable using PODEM.

Step 1: Goal – sensitize fault. Objective – $g = 1$.
Backtrace – Implication stack – $a = 1$. Implications – $g = 1, m = D, n = 0$.
D-frontier – p.

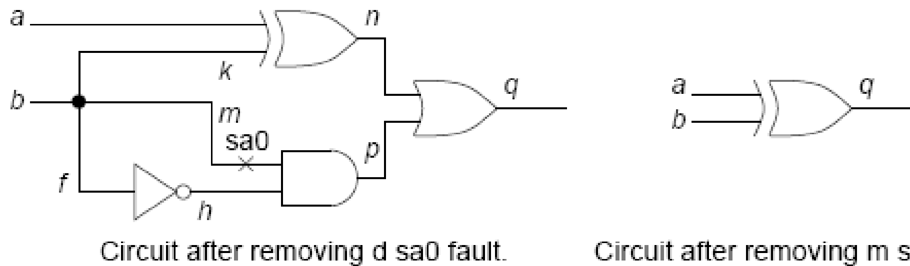
Step 2: Goal – propagate fault. Objective – $h = 1$.
Backtrace – Implication stack – $a = 1, b = 0$. Implications – $g = 1, m = D, n = 0, h = 1, p = D, q = D$.
D-frontier – ϕ *fault at PO.*
Test found – $a = 1, b = 0$; $q = D$.



(3) Redundancy removal.

A. Start with redundant fault d s-a-0.

B. Set fault site to the faulty state and find all implications. For $d = 0$, we find $g = b$. Thus, OR gate g is removed and k and m become fanouts of PI b. The *reduced* circuit is shown on the left in the following figure.



Circuit after removing d sa0 fault. Circuit after removing m sa0 fault

C. Examine the reduced circuit for another redundant fault. We find that m s-a-0, which was testable in the original circuit, is now redundant.

D. Repeat steps B and C until all faults in the reduced circuit are testable.

The above procedure leads to the circuit, $q = a \oplus b$, as shown on the right in the above figure.

Note: This procedure removes only one redundant fault at a time and requires repeated use of ATPG. It is possible to remove several redundant faults together, provided they are selected such that the circuit function is preserved. Removal of a single redundant fault leaves the circuit function unchanged.

7.26 Static Compaction (10 points)

Forward order	Reverse order
$t_1 \cap t_2 = 1010$	$t_5 \cap t_4 = 1100$
$t_3 \cap t_4 = 0100$	$t_3 \cap t_1 = 0010$
$t_5 = 1100$	$t_5 \cap t_4 \cap t_2 = 1100$

Compacted vector sets:
Forward order compaction: 1010, 0100, 1100
Recerse order compaction: 0010,1100
Reverse order is better, as it gives 1 fewer vectors.

(Bushnell and Agrawal) Problem 8.2

It requires just one vector to initialize the circuit. If the initial state is unknown, i.e., $C_n = X$, the vector $A_n = B_n = 1$ initializes the state to 1, irrespective of the presence of any fault at the output S_n. Given this state, detection of any output fault at the output reduces to a combinational ATPG problem of setting the output to the opposite value. This can be done by a single vector: $(A_n = 0, B_n = 0)$ will set the output to 1 or $(A_n = 0, B_n = 1)$ will set it to 0. Thus, just two vectors, an initialization vector 11 followed by an appropriate vector to set the output, will detect the output fault in the circuit of Figure 8.3 (see page 215 of the book.)

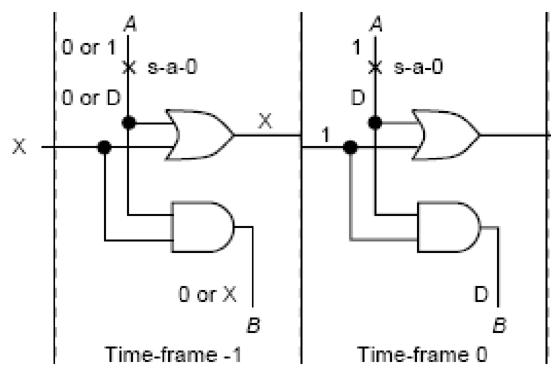
Problem 8.5

For test generation with the five-valued algebra, we use the following steps (also see the illustration):

Step 1: Place a D at the output B in time-frame 0.

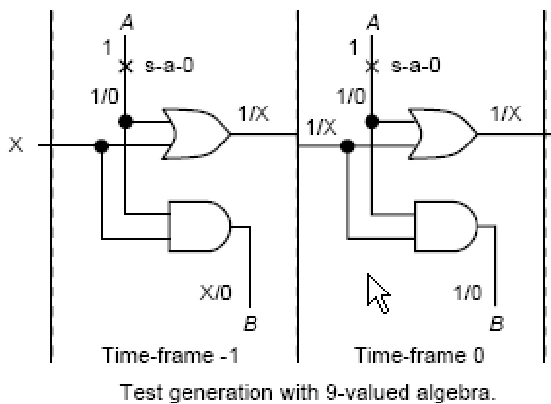
Step 2: This can only be justified by either DD or D1 input to the AND gate in time-frame 0. DD is not possible due to the state input being X in the time-frame -1. We place D1 by applying $A = 1$ and assuming that a state 1 can be justified.

Step 3: Any input, 0 or 1, as shown in the figure, produces a state output X from time-frame -1. Thus, the faulty circuit cannot be initialized to any known state, including the 1 needed for the test. **Hence, it is impossible to find a test by the 5-valued algebra.**



Test generation attempted with 5-valued algebra.

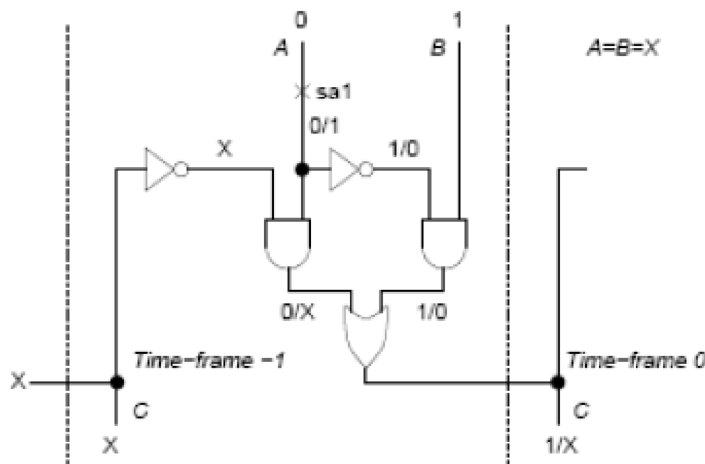
Following similar steps with the nine-valued algebra (see illustration below), we find that two 1's at A detect the fault at B as 1/0 in time-frame 0. Notice that the fault is detected although the faulty circuit is never initialized.

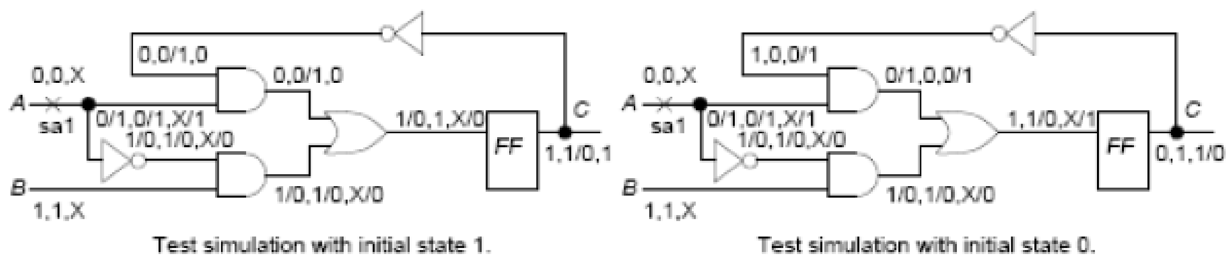


Test generation with 9-valued algebra.

8.6 Initialization fault

The following figure illustrates the time-frame expansion procedure of generating a vector, $A = 0$, $B = 1$, which starting from the unknown state detects the fault A s-a-1 as $1/X$. After the application of the input vector, the flip-flop is clocked before the output can be observed. Even if we add more vectors to the test sequence, the faulty circuit output will not become deterministic. This is because the faulty circuit is not initializable. The fault is only potentially detectable.





Test simulation with initial state 1. Test simulation with initial state 0.

Note: Some test generators will find the potential detection test of the above type. Others will consider the fault untestable (conservative approach.) Most fault simulators will find the fault potentially detectable. Interestingly, the two test simulation scenarios in the figure show that the fault is definitely detectable, though the detection requires multiple observations. If we assume the initial state to be 1 then the fault is detected as 1/0 after the application of the first clock. However, this output will be 1 (same as the correct output) if the initial state was 0. In this case, repeating the same vector and clocking once again will produce a 1/0 output. A conventional fault simulator will not report such detection because it does not enumerate the possible initial state scenarios. For such multiple observation tests see reference [525] of the book.

8.7

The *note* in the solution of Problem 8.6 explains the operation of a multiple observation test. Besides simulation, a multiple observation test can also be derived by the following procedure.

An observable state variable, which cannot be initialized in the faulty circuit but must be observed for fault detection, is represented symbolically by a Boolean variable s. Inversion of s is $\overline{s}$. A test sequence is derived such that any one of the following pairs of outputs is produced:

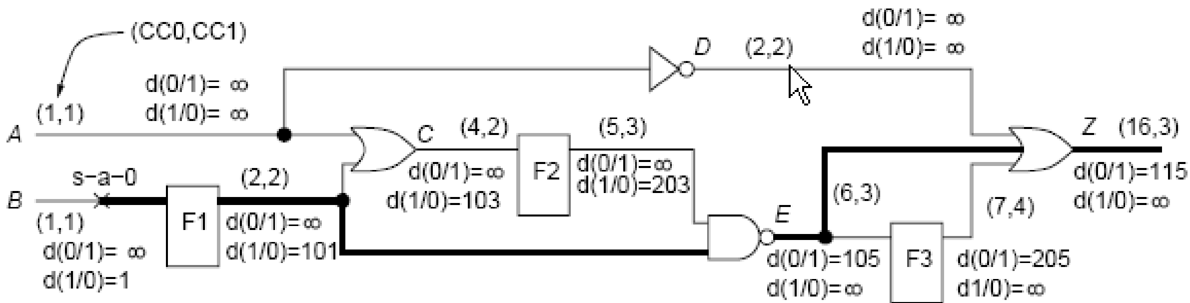
- $0/s$ and $0/\overline{s}$
- $1/s$ and $1/\overline{s}$
- $0/s$ and $1/s$
- $0/\overline{s}$ and $1/\overline{s}$

We notice that irrespective of the value the uninitialized state variable assumes, one element in each test output pair will provide definite fault detection. For example, the outputs produced by the test $(A, B) = (0,1)$, $(0,1)$ of Problem 8.6 are $1/\overline{s}$ and $1/s$, respectively, which agree with the second pair given above.

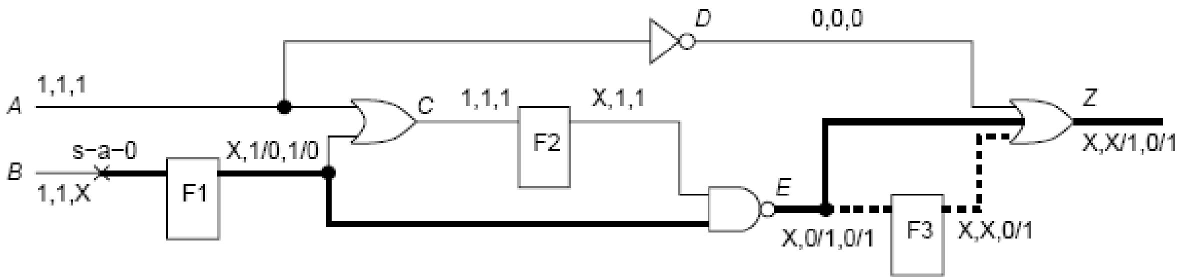
When the feedback in the circuit of Figure 8.25 (see page 250 of the book) has no inversion, a test sequence $(A, B) = (0,0)$, $(0,1)$ will produce outputs $0/s$ and $1/s$. This is a multiple observation test. Details on multiple observation tests may be found in reference [525] cited in the book.

Problem 8.8

The following figure shows the combinational 0 and 1 controllabilities as $(CC0, CC1)$. Notice that the output measures for a flip-flops are obtained by just adding 1 to the input measures. This is due to assumptions that the clock has controllabilities (1,1) and the combinational depth of a flip-flop is 0. The fault site can be driven to 1/0 by controlling $B = 1$ and it cannot be driven to 0/1. Thus, its drivabilities are $d(0/1) = \infty$ and $d(1/0) = 1$, respectively. Drivabilities of all other signals are successively computed by simple path sensitization.



Drivabilities for fault B s−a−0 in circuit of Figure 8.9. Bold lines show easiest drivability path.

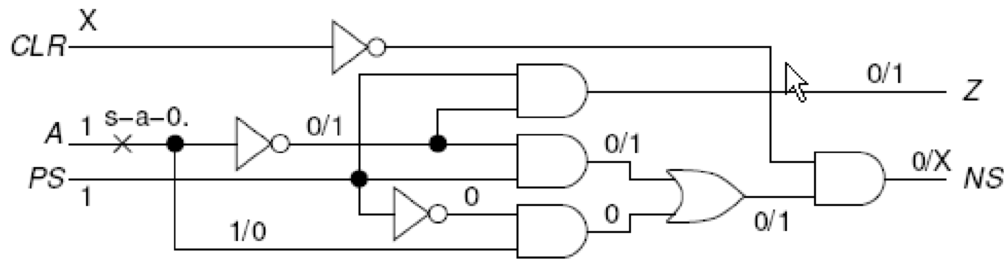


A three−vector test for fault B s−a−0. Dotted lines show an additional path sensitized.

The path shown in bold lines is the least drivability (minimum effort) path. A test obtained by a drivability-based ATPG procedure is shown in the lower figure. This three-vector test, $(A, B) = (1, 1), (1, 1), (1, X)$, sensitizes the minimum drivability path and we find that another path, shown by dotted lines, must also be sensitized.

Problem 8.9 Approximate test

A combinational test for the fault A s-a-0, as shown in the following figure, is $CLR = X,\ A = 1,\ PS = 1$. The fault is detected at Z as $0/1$.

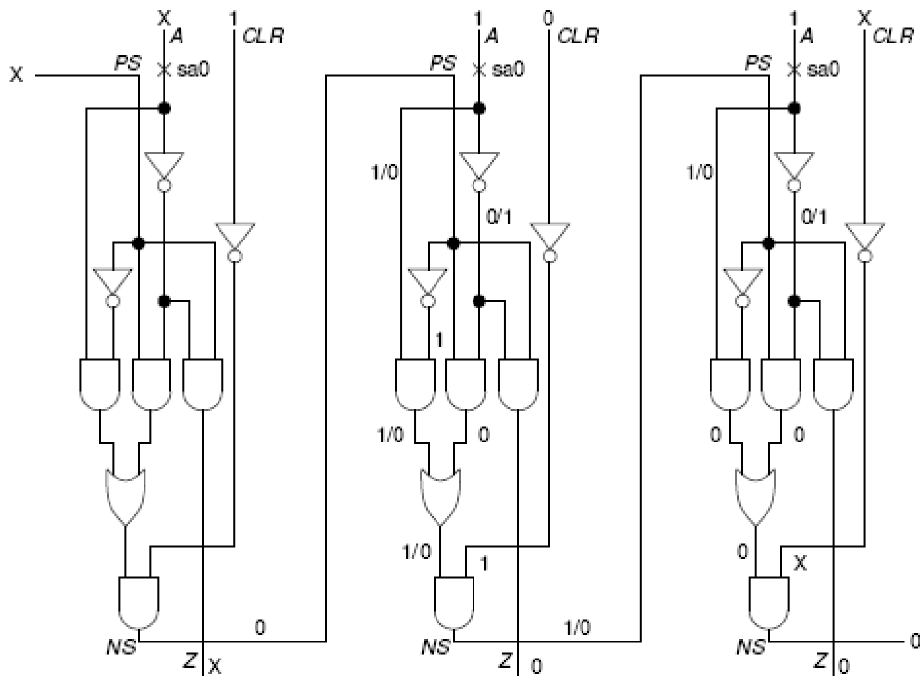


Combinational test for A s–a–0.

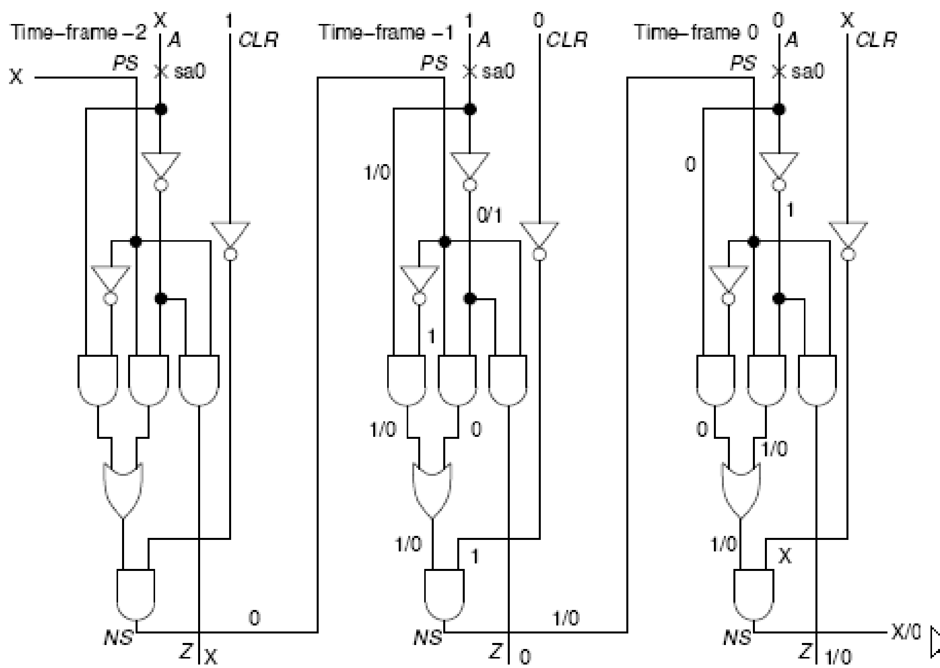
To justify $PS = 1$ in this test, we generate an input vector for the combinational circuit that will produce $NS = 1$ output. We find a vector, $CLR = 0,\ A = 1,\ PS = 0$. In order to apply the required approximation, we assume no fault during justification. The justification must continue until we can find a vector with $PS = X$. $PS = 0$ is easily justified by an input, $CLR = 1,\ A = X,\ PS = X$. Thus, the test sequence contains three vectors, $(CLR, A, PS) = (1, X, X)$, $(0, 1, 0)$, $(X, 1, 1)$, which is simulated in the next figure. We find that the test fails to detect the fault. In the last time-frame, where the combinational vector is applied, the PS input is $1/0$ instead of 1. This is due to the fault being present in the previous time-frame. Thus the faulty previous state interferes with the newly generated fault effect and the output Z becomes 0 instead of $0/1$.

A valid test is generated by time-frame expansion when the fault is assumed to be present in all time-frames (as we did for simulation in the above figure.) The new test, as shown in the following figure, has only one change. In the last time-frame A is changed to 0. So, no new fault effect is produced there and the fault effect $1/0$ produced in time-frame -1 is propagated to Z.

The test sequence is $(CLR, A, PS) = (1, X, X)$, $(0, 1, 0)$, $(X, 0, 1/0)$.



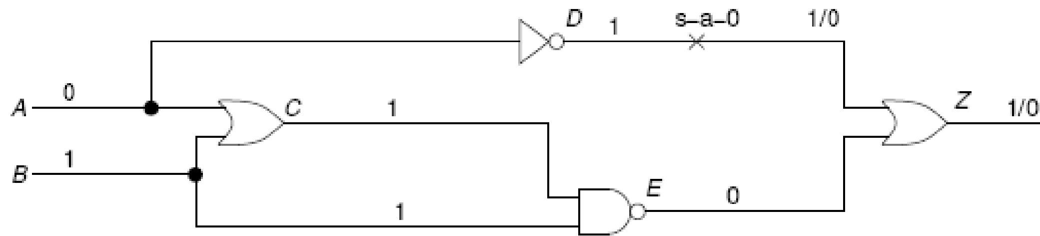
Simulation of approximate test sequence shows it to be be invalid.



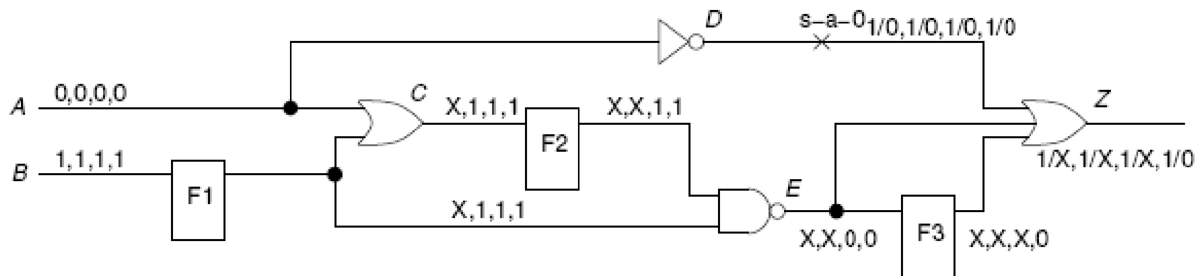
Correct test generation by time−frame expansion method.

Problem 8.12 Pseudo-combinational circuit

The pseudo-combinational circuit and a combinational test, $A = 0,\ B = 1$, for the fault D s-a-0 are shown in the following figure. Simulation of the sequential circuit with input $A = 0,\ B = 1$, repeated four times shows that the fault will be detected as 1/0 appearing as the fourth output. We assume that the initial states of all three flip-flops are X.



Pseudo–combinational circuit for the sequential circuit of Figure 8.9..



Test simulation in sequential circuit.

Problem 8.18 Simulation-based initialization

The initialization sequence for the circuit of Figure 8.9 (see page 226 of the book) is, $(A, B) = (0,0), (1,0)$. The procedure is illustrated in the following table where the selected vectors are shown in boldface.

Simulation-based initialization of circuit of Figure 8.9								
Phase	Types of vectors	Trial vectors		States			Cost	Remarks
		A	B	$F1$	$F2$	$F3$	func.	
I	Initial condition	X	X	X	X	X	3	Cost=#FFs
	Starting vector	**0**	**0**	0	X	X	2	Cost red.
	Unit Hamm. dist.	**1**	**0**	0	1	1	0	Cost red.
	Circuit initialized (cost=0), Phase I completed.							

The initialization sequence for the circuit of Figure 8.13 (see page 230 of the book) is, (CNT, CLR)= (0,1). The procedure is illustrated in the following table where the selected vector is shown in boldface.

Simulation-based initialization of circuit of Figure 8.13							
Phase	Types of vectors	Trial vectors		States		Cost	Remarks
		CNT	CLR	$FF1$	$FF2$	func.	
I	Initial condition	X	X	X	X	2	Cost=#FFs
	Starting vector	0	0	X	X	2	No cost red.
	Unit Hamm. dist.	1	0	X	X	2	No cost red.
		0	**1**	0	0	0	Cost red.
	Circuit initialized (cost=0), Phase I completed.						

This procedure cannot initialize the circuit in Figure 8.12, because neither $CNT = 0$ nor $CNT = 1$ can force any flip-flop into a defined state. These are the only possible trial vectors. Thus, the initial cost of 2 will never be reduced.