1 Introduction

Problem 12.1: **Machine identification.**

Question 1 The states s_1 and s_3 differ ($s_3 \neq s_1$) and are called A and C, respectively. The part of the sequence between s_3 and s_6 must contain all 4 states. Thus, the rightmost part of the state transition table $\mathcal{T}_1$ is completely determined.

I_1 is a homing sequence, as the final state is uniquely determined. O_1 is circular. Therefore, I_1 cannot be a synchronizing sequence.

I_1 could not be shorter. $\mathcal{T}_{1a}$ corresponds to O_{1a}=(0010000).

t	1	2	3	4	5	6	7
I_1	1	1	1	1	1	1	1
O_1	0	0	1	0	0	0	1
s_t	A	B	C	D	A	B	C

$\mathcal{T}_1$	i=0	i=1
A		B,0
B		C,0
C		D,1
D		A,0

$\mathcal{T}_{1a}$	i=0	i=1
A		B,0
B		C,0
C		D,1
D		D,0

Question 2 s_8=D is immediately established. The following three 1's are necessary to determine the next state (D). Application of more 1's does not add to the information gathered.

t	1 2 3 4 5 6 7	8	9	10	11	12	13	14
I_2	1 1 1 1 1 1 1	0	1	1	1	1	1	1
O_2	0 0 1 0 0 0 1	1	0	0	0	1	0	0
s_t	A B C D A B C	D	D	A	B	C	D	A

$\mathcal{T}_1$	i=0	i=1
A		B,0
B		C,0
C		D,1
D	D,1	A,0

Question 3 First, s_{12}=C yields the $i = 0$-input for C.

Next, s_{15}=A yields the $i = 0$-input for A.

Finally, s_{21}=B yields the $i = 0$-input for B.

The state transition table $\mathcal{T}_1$ is determined.

$\mathcal{T}_1$	i=0	i=1
A	A,0	B,0
B	B,0	C,0
C	C,0	D,1
D	D,1	A,0

t	1 2 3 4 5 6 7	8 9 10 11	12	13	14	15	16	17	18	19	20	21	22	23
I_3	1 1 1 1 1 1 1	0 1 1 1	0	1	1	0	1	1	1	1	1	0	1	1
O_3	0 0 1 0 0 0 1	1 0 0 0	0	1	0	0	0	0	1	0	0	0	0	1
s_t	A B C D A B C	D D A B	C	C	D	A	A	B	C	D	A	B	B	C

Problem 12.1: **continued.**

Question 4 3 different machines R_{4a}, R_{4b}, and R_{4c} are considered.

t	1	2	3	4	5	6	7
I_1	1	1	1	1	1	1	1
O_{4a}	0	0	1	0	0	1	0
s_t	?	B	C	A	B	C	A

$\mathcal{T}_{4a}$	i=0	i=1
A		B,0
B		C,0
C		A,1
D		

t	1	2	3	4	5	6	7
I_1	1	1	1	1	1	1	1
O_{4b}	0	0	0	1	1	1	1
s_t	A	B	C	D	D	D	D

$\mathcal{T}_{4b}$	i=0	i=1
A		B,0
B		C,0
C		D,0
D		D,1

t	1	2	3	4	5	6	7
I_1	1	1	1	1	1	1	1
O_{4c}	0	0	0	0	0	0	0
s_t	A	B	C	D	A	B	C

$\mathcal{T}_{4c}$	i=0	i=1
A		?,0
B		
C		
D		

I_1 is a homing sequence for R_{4a} and R_{4b}. I_1 is a synchronizing sequence for R_{4b}, not for R_{4a}. I_1 could be shorter. For R_{4a}, i_7 gives no new information. For R_{4b} and R_{4c} both i_6 and i_7 are superfluous.

For R_{4c}, almost no statements can be made.

Question 5 The only possible deductions about R_5 are shown below. However, many different possibilities exist.

t	1	2	3	4	5	6	7	8
I_5	1	1	1	1	0	0	0	0
O_5	0	0	0	0	0	0	0	0
s_t	?	?	?	A	?	?	?	?

$\mathcal{T}_5$	i=0	i=1
A	?,0	?,0
B		
C		
D		

Question 6 The shortest possible homing sequence $I_6 = (0, 1, 0)$ is constructed for R_6 using exhaustive search.

$$
\begin{tikzcd}
 & & (\alpha,\beta,\gamma,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\alpha,\beta)(\delta,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & & (\alpha,\beta,\gamma,\delta)\ \dagger \\
(\alpha,\beta)(\delta,\delta)\ \dagger & & (\beta,\delta)(\gamma,\gamma) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\alpha)(\delta)(\delta,\delta) & & (\beta,\gamma)(\alpha,\alpha)\ \dagger
\end{tikzcd}
$$

After application of I_6, the final state may be determined uniquely regardless of initial state by observing the output sequence.

<u>Problem 12.1:</u> **continued.**

Question 7 The shortest synchronizing sequence I_7 is constructed for R_6. The output sequence is disregarded.

$$\begin{tikzcd}
 & & (\alpha,\beta,\gamma,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\alpha,\beta,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & & (\alpha,\beta,\gamma,\delta)\ \dagger \\
(\alpha,\beta,\delta)\ \dagger & & (\beta,\gamma,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\alpha,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & & (\alpha,\beta,\gamma)\ \dagger \\
(\beta,\delta)\ \dagger & & (\gamma,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\delta) & & (\alpha,\gamma)\ \dagger
\end{tikzcd}$$

After application of $I_7 = (01010)$, the present state is δ.

Question 8 A minimum length distinguishing sequence I_8 for R_6 is constructed.

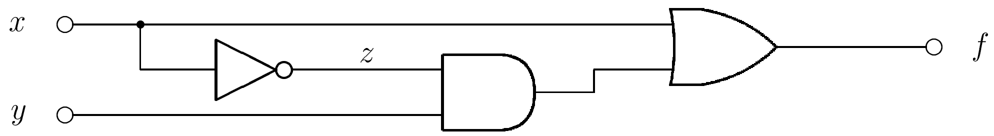
$$\begin{tikzcd}
 & & (\alpha,\beta,\gamma,\delta) \arrow[dl, "0"'] \arrow[dr, "1"] & & & & \\
 & (\beta,\gamma)^0(\alpha)^1\ \dagger & & (\beta)^0(\alpha,\beta,\gamma)^1 \arrow[dl, "0"'] \arrow[drr, "1"] & & & \\
 & & (\gamma)^{00}(\beta,\gamma)^{10}(\alpha)^{11} \arrow[dl, "0"'] \arrow[dr, "1"] & & & (\alpha)^{01}(\beta)^{10}(\gamma,\delta)^{11} \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & (\alpha)^{001}(\gamma)^{100}(\alpha)^{101}(\gamma)^{110} & & (\beta)^{000}(\beta)^{100}(\alpha)^{101}(\delta)^{111} & (\gamma)^{010}(\gamma)^{100}(\beta)^{110}(\alpha)^{111} & & (\delta)^{011}(\alpha)^{101}(\beta)^{110}(\gamma)^{111}
\end{tikzcd}$$

Thus, (100), (101), (110), and (111) are all distinguishing sequences. If, for instance, (101) was selected, the following state sequences would result:

initial state	output sequence			state sequence		
α	1	0	1	δ	β	α
β	1	0	0	α	γ	β
γ	0	0	0	β	γ	β
δ	1	1	1	γ	α	δ

2 Fault Modeling

<u>Problem 12.1:</u> Undetectable faults.



The simplest example is a redundant circuit as shown above. The circuit contains the untestable fault z stuck-at-1. The circuit realises the function $x + y$, where z is an unnecessary literal.

<u>Problem 12.2:</u> Checkpoint faults.

First, note that the following results do not hold for circuits containing EXOR gates.

Question 1 Assume, by contradiction, that a test set $\mathcal{T}$ detects all single stuck faults on primary inputs but does not detect all internal single stuck faults. It suffices to consider internal faults on gate outputs only, as no fanouts are found. There must be a gate $\mathcal{G}$ such that $\mathcal{T}$ detects all faults preceeding $\mathcal{G}$, but not the output fault f.

Assume that $\mathcal{G}$ is an AND gate. Then f can not be a stuck-at-0 fault, since this fault is equivalent to any input stuck-at-0. Similarly, the output stuck-at-1 dominates any input stuck-at-1, and f can not be a stuck-at-1 fault.

Similar arguments hold if $\mathcal{G}$ is a NAND, OR or NOR gate. Thus all the internal faults of the circuit are detected.

Question 2 In the argument above we have to add analysis of all fanout branches. However, it is assumed that all single stuck faults on the fanout branches are detected. Thus, the result is trivial.

<u>Problem 12.3:</u> Checkpoint faults.

For any fault g on a node that is not a checkpoint, at least one checkpoint fault f may be found, such that g dominates f. Thus the number of tests that detect f is smaller than or equal to the number of tests that detect g.

3 Test Pattern Generation

<u>Problem 12.1:</u> Test Setup.

Such a test is *not* possible.

The test vector t will assign some value to $\mathcal{S}$. A test for $\mathcal{S}$ stuck-at-0 requires the value 1 assigned to $\mathcal{S}$, and a test for $\mathcal{S}$ stuck-at-1 requires the opposite value 0. Both can not hold.

<u>Problem 12.2:</u> Boolean differences.

The boolean equations realised by the full-adder are

$$\begin{aligned} s(x_1, x_2, x_3) &= x_1\ x_2\ x_3 + x_1\ \overline{x_2}\ \overline{x_3} + \overline{x_1}\ x_2\ \overline{x_3} + \overline{x_1}\ \overline{x_2}\ x_3 && \text{(sum)} \\ c(x_1, x_2, x_3) &= \overline{x_1}\ x_2\ x_3 + x_1\ \overline{x_2}\ x_3 + x_1\ x_2\ \overline{x_3} + x_1\ x_2\ x_3 && \text{(carry)} \end{aligned}$$

Question 1 First, the carry output is observed. Thus, the boolean difference $\frac{\partial c}{\partial x_1}$ needs to be calculated:

$$\begin{aligned} c_1(0) \equiv c(x_1, x_2, x_3)|_{x_1=0} &= x_2 \cdot x_3 \\ c_1(1) \equiv c(x_1, x_2, x_3)|_{x_1=1} &= x_2 + x_3 \\ \frac{\partial c}{\partial x_1} \equiv\ c_1(0)\ \oplus\ c_1(1) &= x_2\overline{x_3} + \overline{x_2}x_3 \end{aligned}$$

Thus, all tests for x_1 stuck-at-0 observed on the carry output are given by the expression $x_1\ x_2\ \overline{x_3} + x_1\ \overline{x_2}\ x_3$.

All tests for x_1 stuck-at-1 observed on the carry output are given by the expression $\overline{x_1}\ x_2\ \overline{x_3} + \overline{x_1}\ \overline{x_2}\ x_3$.

However, the sum output may also be used. The sum is the exclusive-or of x_1, x_2, and x_3. Thus, the boolean difference $\frac{\partial s}{\partial x_1}$ is identical to 1.

Thus all tests for x_1 stuck-at-0 and x_1 stuck-at-1 observed on the sum output are x_1 and $\overline{x_1}$, respectively.

Question 2 Due to the rotational symmetri, all tests for x_2 stuck-at-0 and stuck-at-1 are x_2 and $\overline{x_2}$, respectively.

Likewise, all tests for x_3 stuck-at-0 and stuck-at-1 are x_3 and $\overline{x_3}$, respectively.

Question 3 A test for s or c stuck-at-0 (stuck-at-1) requires that s or c is set to 1 (0). The conditions are simply the boolean expressions realised by s or c, respectively.

Problem 12.2: continued.

Question 4 The value of ζ may uniquely be determined by observation of c. Thus, all tests for ζ_2 stuck-at-0 and stuck-at-1 are expressed by the terms $c = 0$ and $c = 1$, respectively.

This comprises all input combinations, and no additional information may be obtained from observation of s. However, if the fanout branches of ζ may fail individually, observation of s may become necessary.

Question 5 In order to test for all single stuck faults on x_1 alone, at least two test patterns are required. Thus a minimum of two patterns is required.

Due to the EXOR-nature of the sum output s, any test pattern will simultaneously test three different single stuck faults. In this case it is possible to test all six single stuck faults on the inputs by two test patterns only. 4 different choices of test patterns are found:

$$\begin{array}{ccc} (0,0,0) & \text{and} & (1,1,1) \\ (0,0,1) & \text{and} & (1,1,0) \\ (0,1,0) & \text{and} & (1,0,1) \\ (1,0,0) & \text{and} & (0,1,1) \end{array}$$

Question 6 In order to uniquely identify 1 good and 6 faulty circuits we need $\log_2 7 \approx 2.8$ or 3 bits of information.

With respect to fault detection capabilities, the sum output s dominates the carry output c. No additional information may be obtained from observation of both outputs (this case is somewhat unusual). Thus, at least three test patterns are required.

It is easily verified that selecting 3 test patterns from the first or from the second of the following two sets of test patterns yields a minimal test pattern set for identification:

$$\{(0,0,0),(0,1,1),(1,0,1),(1,0,0)\} \quad \text{or}$$
$$\{(0,0,1),(0,1,0),(1,0,0),(1,1,1)\}$$

Question 7 In order to test for δ stuck-at-0 and ε stuck-at-1, the test patterns $(1,1,1)$ and $(0,0,0)$ are required. These two test patterns will detect all single stuck faults except α, β, γ, ε, or η stuck-at-0.

A test for either α, β, or γ stuck-at-0 requires single path sensitisation through the NOR gate to ζ. The test patterns are $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$, respectively.

Finally, a test pattern detecting ε stuck-at-0 could be either one of the patterns $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. Any of these patterns will also detect η stuck-at-0.

Problem 12.3: Boolean differences.

Question 1 Shannon's expansion theorem applied to $h(x, \boldsymbol{y})$ yields

$$g(h(x,\boldsymbol{y}),\boldsymbol{z}) \;=\; g(0,\boldsymbol{z})\cdot\overline{h(x,\boldsymbol{y})} + g(1,\boldsymbol{z})\cdot h(x,\boldsymbol{y})$$

Using the definition of Boolean differences we get

$$\begin{aligned}
\frac{\partial f(x,\boldsymbol{y},\boldsymbol{z})}{\partial x} &\equiv f(0,\boldsymbol{y},\boldsymbol{z}) \oplus f(1,\boldsymbol{y},\boldsymbol{z}) \\
&= \Big(g(0,\boldsymbol{z})\overline{h(0,\boldsymbol{y})} + g(1,\boldsymbol{z})h(0,\boldsymbol{y})\Big) \oplus \\
&\quad \Big(g(0,\boldsymbol{z})\overline{h(1,\boldsymbol{y})} + g(1,\boldsymbol{z})h(1,\boldsymbol{y})\Big) \\
&= \overline{g(0,\boldsymbol{z})}g(1,\boldsymbol{z})\;\overline{h(0,\boldsymbol{y})}h(1,\boldsymbol{y}) + \overline{g(0,\boldsymbol{z})}g(1,\boldsymbol{z})\;h(0,\boldsymbol{y})\overline{h(1,\boldsymbol{y})} + \\
&\quad g(0,\boldsymbol{z})\overline{g(1,\boldsymbol{z})}\;\overline{h(0,\boldsymbol{y})}h(1,\boldsymbol{y}) + g(0,\boldsymbol{z})\overline{g(1,\boldsymbol{z})}\;h(0,\boldsymbol{y})\overline{h(1,\boldsymbol{y})} \\
&= (g(0,\boldsymbol{z}) \oplus g(1,\boldsymbol{z}))\cdot(h(0,\boldsymbol{y}) \oplus h(1,\boldsymbol{y})) \\
&= \frac{\partial g(h(x,\boldsymbol{y}),\boldsymbol{z})}{\partial h(x,\boldsymbol{y})}\cdot\frac{\partial h(x,\boldsymbol{y})}{\partial x} \qquad\qquad \text{qed}
\end{aligned}$$

Question 2 The relationship is similar to the chain rule known from mathematical analysis. However, a counterexample shows that the relationship does not hold for Boolean functions.

Assume that $G(h,k) = h + k$, $H(x,y) = xy$, and $K(x,z) = \overline{xz}$. Then $F(x,y,z) = xy + \overline{xz}$.

Direct calculation gives the result

$$\frac{\partial F(x,y,z)}{\partial x} \;=\; \overline{y}z$$

Using the stated relationship a different result is obtained

$$\frac{\partial G(H(x,y),K(x,z))}{\partial H(x,y)}\cdot\frac{\partial H(x,y)}{\partial x} + \frac{\partial G(H(x,y),K(x,z))}{\partial K(x,z)}\cdot\frac{\partial K(x,y)}{\partial x}$$

$$= \;\overline{K(x,z)}\cdot y + \overline{H(x,y)}\cdot z \;=\; z$$

Question 3 The *chain rule* may be proven in different ways.

Method 1: The function may be written as $\varrho(x) = g\cdot(p + \sigma(x))$. According to the chain rule, the following should be calculated

$$\frac{\partial \varrho(x)}{\partial \sigma(x)} \;=\; \varrho\big|_{\sigma=0} \oplus \varrho\big|_{\sigma=1} \;=\; (gp)\oplus g \;=\; g\overline{p}$$

Method 2: According to the definition the Boolean difference $\frac{\partial\varrho}{\partial\sigma}$ describes when the value of σ uniquely determines the value of ϱ.

Creation of a *sensitised path* from σ to ϱ requires that $p = 0$ and $g = 1$. The same result is achieved by inspection.

Problem 12.3: **continued.**

Question 4 The set of all tests for c_0 stuck-at-0 and stuck-at-1 are given by the expressions $c_0 \cdot \frac{\partial c}{\partial c_0}$ and $\overline{c_0} \cdot \frac{\partial c}{\partial c_0}$, respectively. Using the chain rule yields

$$\begin{aligned}\frac{\partial c}{\partial c_0} &= \frac{\partial c_4}{\partial c_3} \cdot \frac{\partial c_3}{\partial c_2} \cdot \frac{\partial c_3}{\partial c_2} \cdot \frac{\partial c_2}{\partial c_1} \cdot \frac{\partial c_1}{\partial c_0} \\ &= \overline{p_4}\ \overline{p_3}\ \overline{p_2}\ \overline{p_1}\ g_4\ g_3\ g_2\ g_1\end{aligned}$$

Alternatively, $\frac{\partial c}{\partial c_0}$ may be determined by inspecting the requirements for creation of a sensitised path from c_0 to c. Or the result may be achieved directly by calculation from the logic equation for c.

The only test vectors detecting c_0 stuck-at-0 and stuck-at-1 are
$c_0\ \overline{p_1}\ \overline{p_2}\ \overline{p_3}\ \overline{p_4}\ g_1\ g_2\ g_3\ g_4$ and $\overline{c_0}\ \overline{p_1}\ \overline{p_2}\ \overline{p_3}\ \overline{p_4}\ g_1\ g_2\ g_3\ g_4$, respectively.

Question 5 The boolean differences are given by (for $i = 1, 2, 3, 4$)

$$\frac{\partial c}{\partial p_i} = \frac{\partial c}{\partial c_i} \cdot \frac{\partial c_i}{\partial p_i} = \frac{\partial c}{\partial c_i} \cdot g_i\ \overline{c}_{i-1}$$

$$\frac{\partial c}{\partial g_i} = \frac{\partial c}{\partial c_i} \cdot \frac{\partial c_i}{\partial g_i} = \frac{\partial c}{\partial c_i} \cdot (p_i + c_{i-1})$$

Question 6 All nodal stuck-at faults detected by the four test vectors $t_1 \ldots t_4$ are underlined in the simulation table below. If a node has the value 0 (1), the test vector detects the node stuck-at-1 (stuck-at-0), if and only if a sensitised path exists between the node and the output, c.

	c_0	p_1	p_2	p_3	p_4	g_1	g_2	g_3	g_4	k_1	k_2	k_3	k_4	c_1	c_2	c_3	c
t_1	$\underline{1}$	0	0	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
t_2	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	1	1	1	1	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1}$
t_3	0	1	1	$\underline{0}$	$\underline{0}$	0	$\underline{0}$	1	1	1	1	$\underline{0}$	$\underline{0}$	0	$\underline{0}$	$\underline{0}$	$\underline{1}$
t_4	0	1	$\underline{1}$	0	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	1	$\underline{1}$	$\underline{1}$	$\underline{1}$	0	$\underline{1}$	$\underline{1}$	$\underline{0}$
sa-0	$\surd$		$\surd$			$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
sa-1	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$			$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

The table shows, that 28 out of 34 faults are detected by the 4 vectors.

Question 7 The circuit is fanout-free, thus the test needs only to detect all single stuck-at faults on the inputs.

c_0 stuck-at-0 and stuck-at-1 are detected by the vectors t_1 and t_2 only. In addition, the vectors also detect all faults g_i stuck-at-0 and p_i stuck-at-1.

A test for g_i stuck-at-1 requires $g_i = 0,\ k_i = 1,\ g_{i+1\ldots} = 1$, and $p_{i+1\ldots} = 0$.
4 vectors of a form similar to t_3 are required.

Testing p_i stuck-at-0 requires $c_i = 0,\ g_{i\ldots} = 1,\ p_i = 1$, and $p_{i+1\ldots} = 0$.
Only a single p_i stuck-at-0 fault may be detected by each vector. Four vectors similar to t_4 are required.

A minimal set of test patterns consists of 10 test vectors.

<u>Problem 12.4:</u> Test generation.

The circuit contains redundancy. The fault-free function is

$$z \;=\; \overline{x_1}\;\overline{x_2}\;x_3\;\overline{(x_3 x_4)} \;=\; \overline{x_1}\;\overline{x_2}\;x_3\;x_4$$

Question 1 G_1 is replaced by $\overline{x_1 \overline{x_3}}$. The faulty function is $\overline{x_2}\;x_3\;x_4$.

Question 2 The faulty function is $\overline{x_1}\;x_4$.

Question 3 None of the test vectors detect x_1 stuck-at-0.

The first vector does not excite the fault. The next three vectors do excite the fault, but the fault effect is not propagated to z. The latter requires $\overline{x_2}\;x_3\;x_4$. The only test vector detecting the fault x_1 stuck-at-0 is (1011).

Question 4 All tests detecting the fault x_3 stuck-at-0 are

$$x_3 \cdot \frac{\partial z}{\partial x_3} \;=\; x_3 \cdot [0 \oplus \overline{x_1}\;\overline{x_2}\;x_4] \;=\; \overline{x_1}\;\overline{x_2}\;x_3\;x_4$$

Question 5 All tests detecting the fault x_2 stuck-at-0 are

$$x_2 \cdot \frac{\partial z}{\partial x_2} \;=\; x_2 \cdot [\overline{x_1}\;x_3\;x_4 \oplus 0] \;=\; \overline{x_1}\;x_2\;x_3\;x_4$$

Question 6 All tests detecting the fault x_2 stuck-at-1 are

$$\overline{x_2} \cdot \frac{\partial z}{\partial x_2} \;=\; \overline{x_2} \cdot [\overline{x_1}\;x_3\;x_4 \oplus 0] \;=\; \overline{x_1}\;\overline{x_2}\;x_3\;x_4$$

<u>Problem 12.5:</u> Fault diagnosis.

Question 1 β only affects the output z_2, $z_2 = x_2x_3 + x_2x_3\beta$. β is clearly a redundant signal. No test can be found for β stuck-at-1.

$$\overline{\beta} \cdot \frac{\partial z_2}{\partial \beta} \;=\; \overline{x_1} \cdot [x_2x_3 \oplus x_2x_3] \;=\; 0$$

Question 2 The faulty functions expressed in terms of α and γ are

α stuck-at-0: $\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \overline{x_1} + \alpha x_3 \\ \alpha x_3 + x_1x_2x_3 \end{pmatrix}$

$$\alpha \cdot \frac{\partial \mathbf{z}}{\partial \alpha} \;=\; x_2 \cdot \begin{pmatrix} \overline{x_1} \oplus (\overline{x_1} + x_3) \\ x_1x_2x_3 \oplus x_3 \end{pmatrix} \;=\; \begin{pmatrix} x_1x_2x_3 \\ \overline{x_1}x_2x_3 \end{pmatrix}$$

γ stuck-at-0: $z_1 \;=\; \overline{x_1} + \gamma$

$$\gamma \cdot \frac{\partial z_1}{\partial \gamma} \;=\; \gamma \cdot (\overline{x_1} \oplus 1) \;=\; x_1x_2x_3$$

The faults may be distinguished on the output z_2 by the vector $\overline{x_1}x_2x_3$. On z_1, the faults are equivalent and can not be distinguished.

Problem 12.5: continued.

Question 3 The faulty functions under the presence of the double faults are

α stuck-at-0, β stuck-at-1: $\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \overline{x_1} + \alpha x_3 \\ \alpha x_3 + x_1 x_2 \beta \end{pmatrix}$

β stuck-at-1, γ stuck-at-0: $\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \overline{x_1} \\ x_2 x_3 \end{pmatrix}$

Calculations similar to those under the previous question may be carried out. However, the fault β stuck-at-1 results in a redundant circuit for the output z_2. Under these conditions, the fault effect of α is not visible on z_2. On z_1, the faults are already equivalent. No test can distinguish these double faults.

The interpretation is that the presence of redundancies may invalidate tests for single stuck-at faults.

Problem 12.6: D-notation.

The 5-valued truth table for a two-input exor gate is shown below.

$\oplus$	0	1	X	D	$\overline{D}$
0	0	1	X	D	$\overline{D}$
1	1	0	X	$\overline{D}$	D
X	X	X	X	X	X
D	D	$\overline{D}$	X	0	1
$\overline{D}$	$\overline{D}$	D	X	1	0

Problem 12.7: Test generation algorithms.

Question 1 A test is generated for θ stuck-at-1 using the D-algorithm. In the example, justification is postponed until the D-frontier has reached the output.

Step		α	β	γ	δ	ε	ζ	θ	η	ι	κ	λ	μ	decision
setup	*pcdf*	0	X	-	-	-	-	$\overline{D}$	-	-	-	-	-	√
propagate	pdc_1							$\overline{D}$		0	$\overline{D}$			
	tc_1	0						$\overline{D}$		0	$\overline{D}$			
propagate	pdc_2										$\overline{D}$	1	$\overline{D}$	
	tc_2	0						$\overline{D}$		0	$\overline{D}$	1	$\overline{D}$	
justify λ	sc_3						0					1		
	tc_3	0					0	$\overline{D}$		0	$\overline{D}$	1	$\overline{D}$	
justify ι	sc_4					0			X	0				√
	tc_4	0				0	0	$\overline{D}$		0	$\overline{D}$	1	$\overline{D}$	

No conflicts are encountered. The test vector generated is (0XXX00).

Problem 12.7: continued.

Question 2 The same test is generated using the Podem algorithm as described in the original article by Goel.

	Objective	α	β	γ	δ	ε	ζ	θ	η	ι	κ	λ	μ	
setup	$\overline{D}$ on θ	-	-	-	-	-	-	$\overline{D}$	-	-	-	-	-	
backtrace	0 on α	0	X					0						$\surd$
propagate	0 on κ										0			
backtrace	0 on ε					0				0				$\surd$
propagate	1 on μ												1	
backtrace	0 on ζ						0					1		
imply							0					1	0	
		0				0	0	0		0	0	1	0	

Question 3 The test vector detects all single stuck-at faults along the sensitised path to the output. Very little more may be determined, as half the inputs are unassigned.

In the following, the unassigned inputs are arbitrarily assigned to 0. The fault-free values of the circuit are (000000|010010). In this case, the active faults on α and β as well as γ, δ, and η are not propagated. Similarly, ζ and λ are blocked by κ. However, a sensitised path exists between ε and μ, and all faults activated along this path are detected. The faults detected by the all-zero test vector are ε, θ, ι, κ, and μ stuck-at-1.

Problem 12.8: Test generation algorithms.

Question 1 The D-algorithm is used to generate a test for χ stuck-at-1.

Step		x_1	x_2	x_3	α	β	γ	δ	ε	ζ	χ	η	s	c	
setup	*pcdf*	-	-	-	-	-	-	-	-	1	D	-	-	-	
imply	sc_1									1				0	
propagate	pdc_1								1		D	D			
	tc_1								1	1	D	D		0	
propagate	pdc_2							0				D	D		
	tc_2							0	1	1	D	D	D	0	
justify ζ	sc_3				0	0	0			1					
	tc_3				0	0	0	0	1	1	D	D	D	0	
justify ε	sc_4	1	X	X					1						$\surd$
	tc_4	1			0	0	0	0	1	1	D	D	D	0	
justify δ	sc_{5a}	1	0	X				0							$\surd$
imply	sc_{5b}		0		0		0								
	tc_5	1	0		0	0	0	0	1	1	D	D	D	0	
justify β	sc_6	1		0		0									
	tc_6	1	0	0	0	0	0	0	1	1	D	D	D	0	

No conflicts are encountered. The test vector generated is (000).

Problem 12.8: continued.

Question 2 The Podem algorithm is used to generate a test for the fault χ stuck-at-0. All decisions are indicated.

Objective		x_1	x_2	x_3	α	β	γ	δ	ε	ζ	χ	η	s	c	
setup	D on χ	-	-	-	-	-	-	-	-	-	D	-	-	-	
backtrace	0 on x_1	0			0					1					$\surd$
imply		0			0	0		0							
backtrace	0 on x_2		0				0			1					$\surd$
imply		0					0			1	1			0	
propagate	1 on η											1			
backtrace	1 on x_3			1					1			1			
imply				1					1			1	1		
		0	0	1	0	0	0	0	1	1	1	1	1	0	

Question 3 x_1, x_2, and x_3 are the only free lines and head lines. All other signals are bound lines.

Question 4 The FAN algorithm is used to generate the test. In this case, the unique path sensitisation procedure is used reaching the output in one step. The D value must pass along gates G_7 and G_9 to the output s. The requirements are that $\delta = 0$ end $\varepsilon = 1$ simultaneously (this is called *multiple path backtrace*).

Step	x_1	x_2	x_3	α	β	γ	δ	ε	ζ	χ	η	s	c	
assignment	-	-	-	-	-	-	-	-	-	D	-	-	-	
unique path							0	1		D	D	D		
backtrace	1	0					0	1						$\surd$
implication	1	0		0		0								
trace $\{\zeta, \beta, c\}$			0						1					
implication			0		0				1				0	
test vector	1	0	0	0	0	0	0	1	1	D	D	D	0	

4 Test Pattern Evaluation

Problem 12.1: Parallel fault simulation.

Question 1 Initially, the fault list contains all 24 single stuck faults:

$$\mathcal{FL}_1 = \{x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,0}, x_{4,1}, x_{5,0}, x_{5,1}, \\ \alpha_0, \alpha_1, \beta_0, \beta_1, \gamma_0, \gamma_1, \delta_0, \delta_1, \varepsilon_0, \varepsilon_1, \zeta_0, \zeta_1, y_0, y_1\}$$

$x_{1,0}$ denotes line x_1 stuck-at-0.

Considering checkpoints only, the fault list is reduced to 14 faults:

$$\mathcal{FL}_2 = \{x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,0}, x_{4,1}, x_{5,0}, x_{5,1}, \beta_0, \beta_1, \gamma_0, \gamma_1\}$$

The fault list may be reduced further considering fault equivalence. Clearly, $x_{1,0} \sim x_{2,0}$ and $x_{4,0} \sim \gamma_0$, and $x_{2,0}$ and $x_{4,0}$ are dropped from the fault list.

Considering fault dominance, the fault list may be reduced even further. $(\alpha_1 \sim \beta_1) \ \wedge \ (\alpha_1 \supset x_{1,1})$ allows for dropping β_1. Similarly, $(\varepsilon_1 \sim x_{5,1}) \ \wedge \ (\varepsilon_1 \supset \gamma_1)$ allows for dropping $x_{5,1}$.

The resulting fault list contains only 10 stuck faults:

$$\mathcal{FL} \ = \ \{x_{1,0}, x_{1,1}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,1}, x_{5,0}, \beta_0, \gamma_0, \gamma_1\}$$

Question 2 11 positions of the computer word is used to represent the good and the 10 different faulty machines.

The result of the simulation is shown below.

Node	good	$x_{1,0}$	$x_{1,1}$	$x_{2,1}$	$x_{3,0}$	$x_{3,1}$	$x_{4,1}$	$x_{5,0}$	β_0	γ_0	γ_1	unused
x_1	0	0	1	0	0	0	0	0	0	0	0	.....
x_2	0	0	0	1	0	0	0	0	0	0	0	.....
x_3	1	1	1	1	0	1	1	1	1	1	1	.....
x_4	0	0	0	0	0	0	1	0	0	0	0	.....
x_5	0	0	0	0	0	0	0	0	0	0	0	.....
α	0	0	0	0	0	0	0	0	0	0	0	.....
β	1	1	1	1	0	1	1	1	0	1	1	.....
γ	1	1	1	1	0	1	1	1	1	0	1	.....
δ	0	0	0	0	1	0	0	0	1	0	0	.....
ε	0	0	0	0	0	0	1	0	0	0	0	.....
ζ	0	0	0	0	0	0	1	0	0	0	0	.....
y	0	0	0	0	1	0	1	0	1	0	0	.....

The detected faults of the reduced fault lists are x_3 stuck-at-0, x_4 stuck-at-1, and β stuck-at-0.

Problem 12.2: **Deductive fault simulation.**

The reduced fault list is determined in Problem 4.1:

$$\mathcal{FL} \;=\; \{x_{1,0}, x_{1,1}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,1}, x_{5,0}, \beta_0, \gamma_0, \gamma_1\}$$

Questions 1–2 Fault propagation rules for 2-input AND, OR and NOR gates:

AND: AB	Rules
00	$L_A \cap L_B$
01	$L_A - L_B$
11	$L_A \cup L_B$
10	$L_B - L_A$

OR/NOR: AB	Rules
00	$L_A \cup L_B$
01	$L_B - L_A$
11	$L_A \cap L_B$
10	$L_A - L_B$

Questions 3–8 The sequence of test vectors $t_1 \ldots t_6$ are applied to the circuit. In every step, the circuit is simulated using an event-driven simulator, and the fault lists are updated. The active fault list on the output, y, contains new faults detected in each step. These faults are subsequently dropped from the fault list.

	t_1	active faults	t_2	active faults	t_3	active faults
x_1	**0**	$\rightarrow \{x_{1,1}\}$	0	$\{x_{1,1}\}$	**1**	$\rightarrow \{x_{1,0}\}$
x_2	**0**	$\rightarrow \{x_{2,1}\}$	0	$\{x_{2,1}\}$	0	$\{x_{2,1}\}$
x_3	**1**	$\rightarrow \{x_{3,0}\}$	1	$\{x_{3,0}\}$	1	$\emptyset$
x_4	**1**	$\rightarrow \emptyset$	**0**	$\rightarrow \{x_{4,1}\}$	0	$\emptyset$
x_5	**0**	$\rightarrow \emptyset$	0	$\emptyset$	**1**	$\rightarrow \{x_{5,0}\}$
α	**0**	$\rightarrow \emptyset$	0	$\emptyset$	0	$\rightarrow \{x_{2,1}\}$
β	**1**	$\rightarrow \{x_{3,0}, \beta_0\}$	1	$\{x_{3,0}, \beta_0\}$	1	$\emptyset$
γ	**1**	$\rightarrow \{x_{3,0}, \gamma_0\}$	1	$\{x_{3,0}\}$	1	$\emptyset$
δ	**0**	$\rightarrow \{x_{3,0}, \beta_0\}$	0	$\{x_{3,0}, \beta_0\}$	0	$\rightarrow \emptyset$
ε	**1**	$\rightarrow \{x_{3,0}, \gamma_0\}$	**0**	$\rightarrow \{x_{4,1}\}$	0	$\emptyset$
ζ	**1**	$\rightarrow \{x_{3,0}, \gamma_0\}$	**0**	$\rightarrow \{x_{4,1}\}$	**1**	$\rightarrow \{x_{5,0}\}$
y	**1**	$\rightarrow \{\gamma_0\}$	**0**	$\rightarrow \{x_{3,0}, x_{4,1}, \beta_0\}$	**1**	$\rightarrow \{x_{5,0}\}$

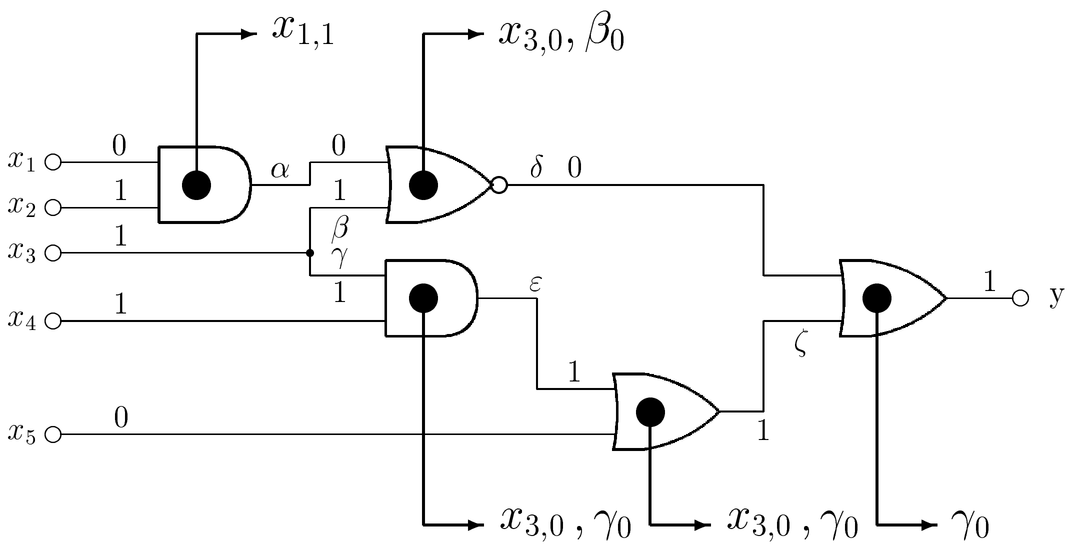
	t_4	active faults	t_5	active faults	t_6	active faults
x_1	1	$\{x_{1,0}\}$	**0**	$\rightarrow \{x_{1,1}\}$	**1**	$\rightarrow \emptyset$
x_2	**1**	$\rightarrow \emptyset$	1	$\emptyset$	**0**	$\rightarrow \{x_{2,1}\}$
x_3	**0**	$\rightarrow \{x_{3,1}\}$	0	$\emptyset$	0	$\emptyset$
x_4	**1**	$\rightarrow \emptyset$	**0**	$\rightarrow \emptyset$	0	$\emptyset$
x_5	**0**	$\rightarrow \emptyset$	0	$\emptyset$	0	$\emptyset$
α	**1**	$\rightarrow \{x_{1,0}\}$	**0**	$\rightarrow \{x_{1,1}\}$	0	$\rightarrow \{x_{2,1}\}$
β	**0**	$\rightarrow \{x_{3,1}\}$	0	$\emptyset$	0	$\emptyset$
γ	**0**	$\rightarrow \{x_{3,1}, \gamma_1\}$	0	$\emptyset$	0	$\emptyset$
δ	0	$\rightarrow \{x_{1,0}\}$	**1**	$\rightarrow \{x_{1,1}\}$	1	$\rightarrow \{x_{2,1}\}$
ε	0	$\rightarrow \{x_{3,1}, \gamma_1\}$	0	$\emptyset$	0	$\emptyset$
ζ	**0**	$\rightarrow \{x_{3,1}, \gamma_1\}$	0	$\emptyset$	0	$\emptyset$
y	**0**	$\rightarrow \{x_{1,0}, x_{3,1}, \gamma_1\}$	**1**	$\rightarrow \{x_{1,1}\}$	1	$\rightarrow \{x_{2,1}\}$

Problem 12.3: **Concurrent fault simulation.**

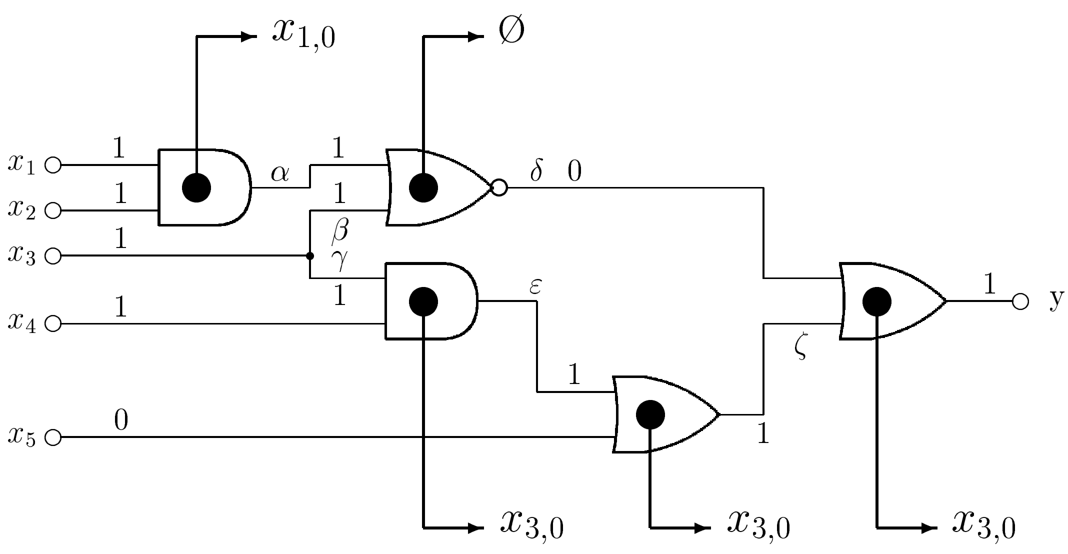
The reduced fault list is determined in Problem 4.1:

$$\mathcal{FL} \;=\; \{x_{1,0}, x_{1,1}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,1}, x_{5,0}, \beta_0, \gamma_0, \gamma_1\}$$

Question 1 First, the test vector (0,1,1,1,0) is applied. γ_0 is detected and consequently dropped from the fault list.



Question 2 Next, the test vector changes to (1,1,1,1,0). The vector detects the fault $x_{3,0}$.



5 Testability Analysis

Problem 12.1: **Ad-hoc design rules.**

Circuit a Double frequency clock generators are hard to verify and seldomly necessary. UART designs incorporate such generators, and special test procedures have to be set up.

Circuit b-c Delay lines as well as the timing parameters from monostables are very sensitive to process variations. Verification is necessary. In synchronous design, the use of delay lines is not justified. Furthermore, hazards may arise from the use of monostables.

Circuit d On-chip oscillators have process-dependant frequencies and belong to the same class as monostables and delay lines. In general, external frequency selection and disable is required for testing.

Circuit e The 3 main disadvantages of RS flip-flops are:
1) unknown return from the illegal input state, 2) asynchronous operation, 3) high spike and glitch sensitivity.
A clocked version is almost always preferable.

Circuit f Implicit flip-flops may be created by feedback loops in combinational circuitry and have the same disadvantages as asynchronous RS flip-flops.

Circuit g Decoders feeding clock inputs may have serious consequences. Due to different lengths of the data paths, and consequently gate delays, decoders are inherently spike ridden and prone to hazards.

Circuit h-i Latches feeding clock inputs are a traditional way of eliminating race hazards and glitches. However, a race condition exists, as the data of the second DFF is sampled at the point of change.
Clock gating does not solve this problem. In this case, spikes may be generated on the gated clock signal.

Circuit j An asynchronously cleared DFF will illustrate problems similar to those of gated clocks as well as the disadvantages of asynchronous RS flip-flops. In the example, the q output pulse is foreshortened, and the DFF is held reset, or unavailable, into the next period.

Circuit k In deferred clock generation, a central clock generator is used to generate a number of secondary, slower clocks. The entire system is vulnerable to clock skew of secondary clocks with respect to primary clocks, thus implying that the primary clock is not available.

Conditionally executed operations have to pass through D-types in order to eliminate spikes and glitches and to bring the signal into synchronization with other signals. Several layers of D-types may be found between each operation, and each layer will have to be tested individually.

<u>Problem 12.2:</u> COEDS Algebra.

Question 1 The 3-input EXOR gate (inputs α, β, γ, output ω) is characterized by the equations

$$\begin{aligned}
KK0(\omega) &= min\,(KK0(\alpha) + KK0(\beta) + KK0(\gamma), KK1(\alpha) + KK1(\beta) + KK0(\gamma),\\
&\quad KK1(\alpha) + KK0(\beta) + KK1(\gamma), KK0(\alpha) + KK1(\beta) + KK1(\gamma)) + 1\\
KK1(\omega) &= min\,(KK1(\alpha) + KK1(\beta) + KK1(\gamma), KK0(\alpha) + KK0(\beta) + KK1(\gamma),\\
&\quad KK0(\alpha) + KK1(\beta) + KK0(\gamma), KK1(\alpha) + KK0(\beta) + KK0(\gamma)) + 1\\
SK0(\omega) &= min\,(SK0(\alpha) + SK0(\beta) + SK0(\gamma), SK1(\alpha) + SK1(\beta) + SK0(\gamma),\\
&\quad SK1(\alpha) + SK0(\beta) + SK1(\gamma), SK0(\alpha) + SK1(\beta) + SK1(\gamma))\\
SK1(\omega) &= min\,(SK1(\alpha) + SK1(\beta) + SK1(\gamma), SK0(\alpha) + SK0(\beta) + SK1(\gamma),\\
&\quad SK0(\alpha) + SK1(\beta) + SK0(\gamma), SK1(\alpha) + SK0(\beta) + SK0(\gamma))\\
KO(\alpha) &= KO(\omega) + min\,(KK0(\beta), KK1(\beta)) + min\,(KK0(\gamma), KK1(\gamma)) + 1\\
KO(\beta) &= KO(\omega) + min\,(KK0(\alpha), KK1(\alpha)) + min\,(KK0(\gamma), KK1(\gamma)) + 1\\
KO(\gamma) &= KO(\omega) + min\,(KK0(\alpha), KK1(\alpha)) + min\,(KK0(\beta), KK1(\beta)) + 1\\
SO(\alpha) &= SO(\omega) + min\,(SK0(\beta), SK1(\beta)) + min\,(SK0(\gamma), SK1(\gamma))\\
SO(\beta) &= SO(\omega) + min\,(SK0(\alpha), SK1(\alpha)) + min\,(SK0(\gamma), SK1(\gamma))\\
SO(\gamma) &= SO(\omega) + min\,(SK0(\alpha), SK1(\alpha)) + min\,(SK0(\beta), SK1(\beta))
\end{aligned}$$

Question 2 The MOS transistors are 3-terminal bidirectional elements. The terminal are called g(ate), s(ource), and d(rain).

The nMOSt is described by the equations

$$\begin{aligned}
KK0(s) &= 1 + KK1(g) + KK0(d)\\
KK1(s) &= 1 + KK1(g) + KK1(d)\\
KK0(d) &= 1 + KK1(g) + KK0(s)\\
KK1(d) &= 1 + KK1(g) + KK1(s)\\
SK0(s) &= SK1(g) + SK0(d)\\
SK1(s) &= SK1(g) + SK1(d)\\
SK0(d) &= SK1(g) + SK0(s)\\
SK1(d) &= SK1(g) + SK1(s)\\
KO(g) &= 1 + KO(s) + KK0(d) + KK1(d)\\
KO(g) &= 1 + KO(d) + KK0(s) + KK1(s)\\
KO(s) &= 1 + KO(d) + KK1(g)\\
KO(d) &= 1 + KO(s) + KK1(g)\\
SO(g) &= SO(s) + SK0(d) + SK1(d)\\
SO(g) &= SO(d) + SK0(s) + SK1(s)\\
SO(s) &= SO(d) + SK1(g)\\
SO(d) &= SO(s) + SK1(g)
\end{aligned}$$

<u>Problem 12.2:</u> continued.

Similarly, the pMOSt is described by the equations

$$\begin{aligned}
KK0(s) &= 1 + KK0(g) + KK0(d)\\
KK1(s) &= 1 + KK0(g) + KK1(d)\\
KK0(d) &= 1 + KK0(g) + KK0(s)\\
KK1(d) &= 1 + KK0(g) + KK1(s)\\
SK0(s) &= SK0(g) + SK0(d)\\
SK1(s) &= SK0(g) + SK1(d)\\
SK0(d) &= SK0(g) + SK0(s)\\
SK1(d) &= SK0(g) + SK1(s)\\
KO(g) &= 1 + KO(s) + KK0(d) + KK1(d)\\
KO(g) &= 1 + KO(d) + KK0(s) + KK1(s)\\
KO(s) &= 1 + KO(d) + KK0(g)\\
KO(d) &= 1 + KO(s) + KK0(g)\\
SO(g) &= SO(s) + SK0(d) + SK1(d)\\
SO(g) &= SO(d) + SK0(s) + SK1(s)\\
SO(s) &= SO(d) + SK0(g)\\
SO(d) &= SO(s) + SK0(g)
\end{aligned}$$

Question 3 The COEDS algorithm is an iterative algorithm that processes the contents of a *gate stack* much like an event-driven simulator. The algorithm has no built-in direction of signal flow, and a terminal is not marked as neither input nor output. Thus, bidirectional elements are easily handled by the COEDS algorithm.

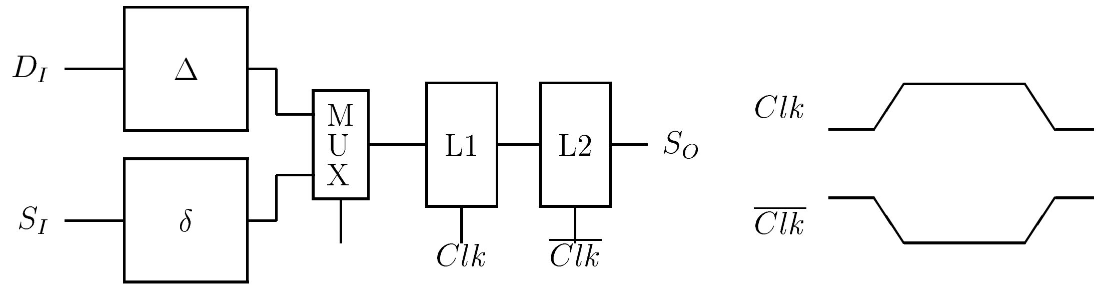
<u>Problem 12.3:</u> Design review.

The circuit may effectively be partitioned into three parts, the upper, middle and lower part. These parts may be tested in various orders.
The circuit is easily initialised. The upper counter may be reset by clocking twice while holding CLKRST at 0. The middle counter (IC2) may be loaded in parallel. Finally, SELA is easily controlled.

The upper counter (IC1s) is easily verified. It is important to note that the state of this counter may be controlled independantly of the rest of the circuit.
The lower part is both controllable and observable, except for signal SELA. This signal may be tested indirectly observing e.g. Y3.
For the middle part, the attention is drawn to Y0 and Y2 (while Y1 and Y3 may easily be tested). The rightmost latch has the illegal input state 00, and the circuit must be verified against this. During the count sequence, (QA1 $\searrow$ 0, QC0 $\nearrow$ 1) occurs simultaneously twice. Detailed Min-Max timing analysis will show a small, but finite risk of a hazard.

6 Design for Testability

Problem 12.1: **Ad-hoc design rules.**



The clock race condition is inherent to two-phase clocking. Clock skew arising from slow clock edges may permit data to ripple through 2 or more latches. If the normal mode delay, Δ, is larger than the scan chain wiring delay, δ, the data on the scan chain are invalidated.

Problem 12.2: **LSSD design.**

The number of clock cycles required is determined from the scan test scenario:

scan-in :	$99 \cdot (A + B)$	
sample :	$(C + B)$	pattern 1
scan-out/scan-in :	$100 \cdot (A + B)$	
sample :	$(C + B)$	pattern 2
	$\cdots$	
sample :	$(C + B)$	$\cdots$
scan-out/scan-in :	$100 \cdot (A + B)$	
sample :	$(C + B)$	pattern 1000
scan-out :	$99 \cdot (A + B)$	
	101.098 clock phases	

Problem 12.3: **LSSD design.**

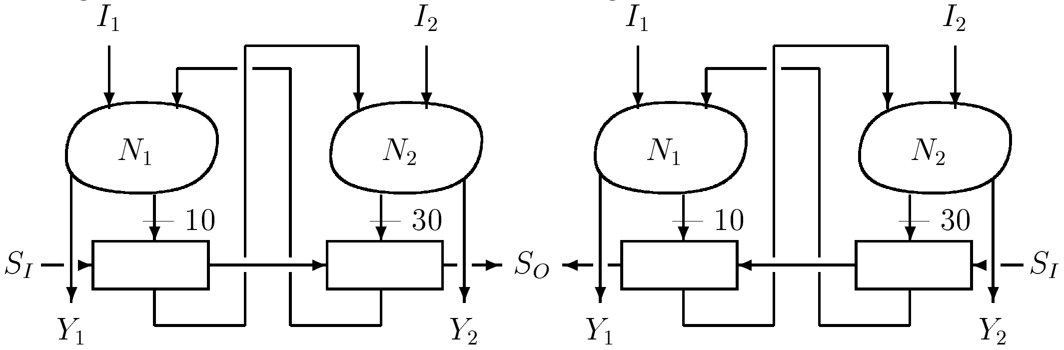
In single-latch LSSD design, the number of secondary outputs of N_1 equals the number of secondary inputs to N_2. The scan path may be arranged in the two configurations shown below. In both configurations, the length of (this part of) the scan path is 40.

If the scan path is operated in the usual way, labeled 'overlap mode', N_1 and N_2 are tested in parallel. Ignoring initial and termination costs, this would require $40 \cdot 30$ scan path shift operations, regardless of the configuration chosen and the actual placement of the test patterns.

Problem 12.3: continued.

However, if the scan path is operated in a slightly non-standard way, considerable savings may be achieved. In this case, 10 test patterns are applied as in 'overlap mode', followed by 20 patterns applied only to the specific logic block, labeled 'separate mode'.

Configuration 1 Configuration 2



Question 1 In Configuration 1, the number of scan chain shift operations are

10 'overlap' patterns	$10 \cdot (30 + 10)$	400
20 'separate' N_2 patterns	$20 \cdot (max\{10, 30\})$	600
Total scan chain shift operations		1.000

In Configuration 2, the number of scan chain shift operations are

10 'overlap' patterns	$10 \cdot (30 + 10)$	400
20 'separate' N_2 patterns	$20 \cdot (max\{40, 40\})$	800
Total scan chain shift operations		1.200

In this case Configuration 1 should be selected.

Question 2 In Configuration 1, the number of scan chain shift operations are

10 'overlap' patterns	$10 \cdot (30 + 10)$	400
20 'separate' N_1 patterns	$20 \cdot (max\{40, 40\})$	800
Total scan chain shift operations		1.200

In Configuration 2, the number of scan chain shift operations are

10 'overlap' patterns	$10 \cdot (30 + 10)$	400
20 'separate' N_1 patterns	$20 \cdot (max\{30, 10\})$	600
Total scan chain shift operations		1.000

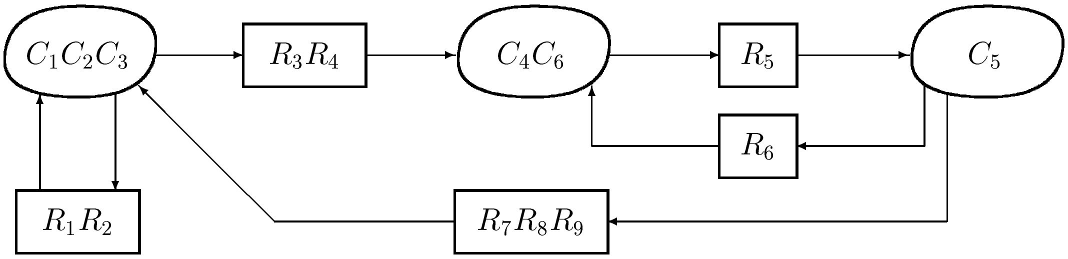
In this case Configuration 2 should be selected.

Problem 12.4: Partial scan design.

In general, introduction of scan paths adds costs, which may only be justified by some testing problem finding a simpler solution. Random selection of latches for partial scan paths may not improve on testing costs.

As a counterexample, consider the partitioning example of Problem 6.6. Here, a number of flip-flops are essential, and a number of others certainly are not.

Problem 12.5: Partial scan design.



The circuit is regrouped into maximal regions of connected combinational logic, such that the inputs to each group are either primary inputs or latch outputs; the outputs are either primary outputs or latch inputs. The regrouped structure is shown in the figure.

After implementation of partial scan, each of the combinational groups should be tested by the application of a test pattern, t_i, and observation of the result, t_o. Each test pattern consists of two parts, where t_i^a is applied to the primary inputs of the group, while t_i^b is applied to the secondary inputs using the scan path. Similarly, the output part consists of t_o^a and t_o^b.

The test proceeds as follows. First, t_i^b is scanned into the scan path. Next, t_i^a is applied. The registers of the group is clocked d times while holding the inputs. d denotes the sequential depth of the group. Then the results t_o^a are observed, and t_o^b is scanned out through the scan path. Using the above procedure, it is essential that the partial scan chains break all feed-back loops.

Identification of flip-flops that must be included into the scan path is straight forward from the regrouped circuit. R_1R_2 are essential due to the self loop. R_3R_4 is chosen over $R_7R_8R_9$ to break the second loop. A choice between R_5 and R_6 must be made to break the third loop; R_5 is chosen arbitrarily. Thus, R_1 through R_5 are put into the scan path.

Problem 12.6: Generator polynomials.

The polynomial is the CCITT international standard. The polynomial itself, $G(x)$, is not irreducible, but is the product of $x+1$ and an irreducible polynomial; Therefore, the polynomial does not produce the optimal aliasing properties (given the length of the polynomial):

$$x^{16} + x^{12} + x^5 + 1 \quad = \quad (x+1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x^1 + 1)$$

Question 1 The condition for aliasing is that the error polynomial is divisible by the generator polynomial, $G(x)$. An error polynomial representing any single error, x^q, is clearly not divisible by the generator polynomial.

Not all double errors are in fact detected. A double error may be represented by $x^p(x^q + 1)$. $G(x)$ is not divisible by x, and it may be shown that $G(x)$ will not divide $x^q + 1$ for $q < 32768$. However, $G(x)$ divides $x^{32768} + 1$, thus two single errors separated 32768 clocks will not be detected.

Problem 12.6: **continued.**

Question 2 If an odd number of error bits are encountered, the error polynomial, $E(x)$, contains an odd number of bits (e.g. $E(x) = (x^q + x^r + 1)x^p$). No polynomial with an odd number of terms is divisible by $x + 1$, and all error sequences with an odd number of bits are detected. The later may be realized by the following argument.

Assume, by contradiction, that $E(x) = (x + 1)Q(x)$. If $E(x)$ has an odd number of terms, substituting 1 for x everywhere gives $E(1) = 1$. However, the assumption also gives $E(1) = (1 + 1)Q(1) = 0$.

Question 3 The error polynomial for an error sequence of length k may be written as $E(x) = x^p(x^{k-1} + \ldots + 1)$. If k is 16 or less, then the degree of $E(x)$ is less than the degree of $G(x)$, and the remainder may not be 0.

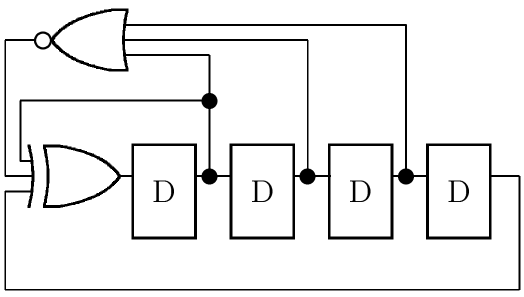
Question 4 If k is 17, then $E(x)$ is divisible by $G(x)$ if and only if $E(x) = G(x)$. From the definition of k, the first and last bits of $E(x)$ are 1, and a match depends on the intermediate 15 bits.

If all combinations are regarded as equally likely, the detection probability is $\frac{1}{2}^{15} \approx 99.997\%$.

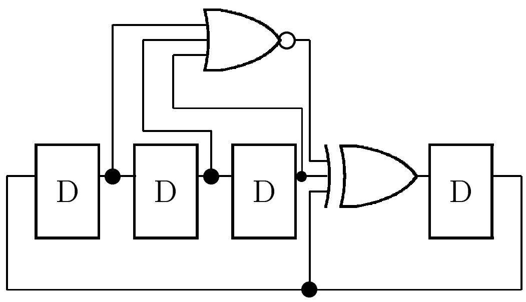
Question 5 If k is 18 or more, the detection probability will depend on the degree of $G(x)$, i.e. $\frac{1}{2^{16}-1} \approx 99.998\%$.

Problem 12.7: **Anti-linear feedback shift registers.**

External-XOR LFSR



Internal-XOR LFSR



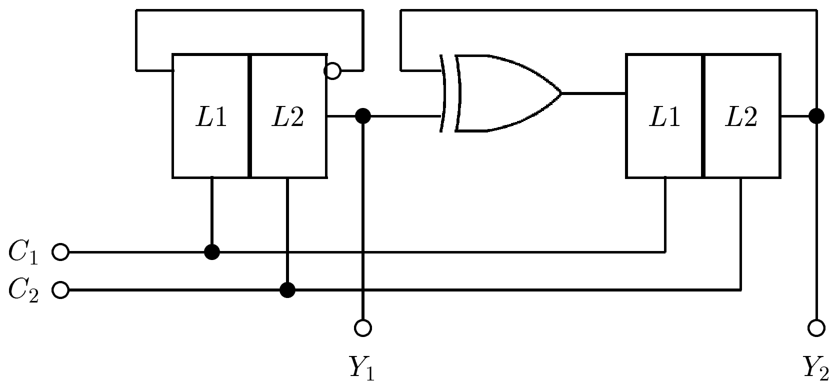
An ALFSR may be constructed from a LFSR by finding a position in the sequence that allows for the insertion of the all-zero pattern. For both types of LFSRs, such positions are indicated in the following table. The required transition may be achieved by an $n-1$-bit wide NOR gate.

LFSR type	External-XOR	Internal-XOR
First pattern	$00\ldots01$	$00\ldots01$
Inserted pattern	$00\ldots00$	$00\ldots00$
Second pattern	$100\ldots0$	$G(x)$

Problem 12.8: **L1L2 design: Clock gating.**

Question 1 The 2-bit binary counter is described by the excitation equations

$$Y_1 = \overline{y_1} \qquad\qquad Y_2 = y_1 \oplus y_2$$



Question 2 The result is trivial for the 'ungated clock' circuit.

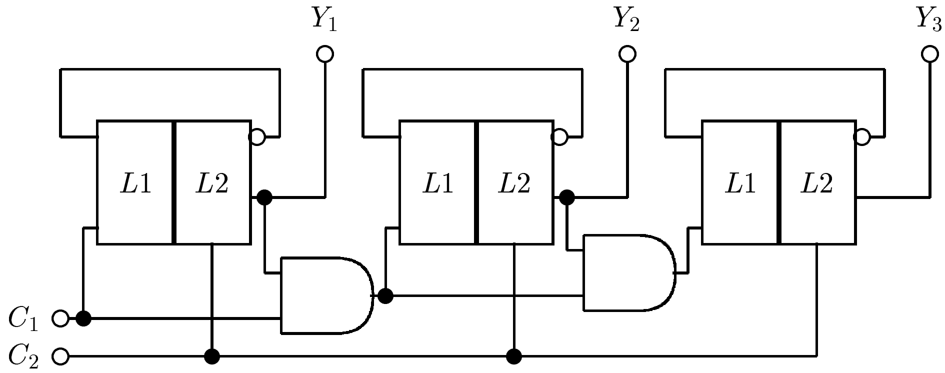
For the 'gated C_1 clock' circuit, f is latched if and only if g is 1; otherwise the circuit holds the present value, y_i.

Question 3 The gated clock circuit is in accordance with the 6 requirements for LSSD design:

1) Polarity-hold latches only are used,
2) The controlling clocks, C_1 and C_2, are independent,
3) C_1 and C_2 may be clock primary inputs,
4) The clocks do not feed the data inputs,
5) A scan chain may readily be constructed, and
6) The scan state may be identified.

Question 4 The 3-bit binary counter is designed using the gated design style. The excitation equations are

$$Y_1 = \overline{y_1} \qquad Y_2 = y_1 \oplus y_2 \qquad Y_3 = (y_1 y_2) \oplus y_3$$



7 Test of Regular Structures

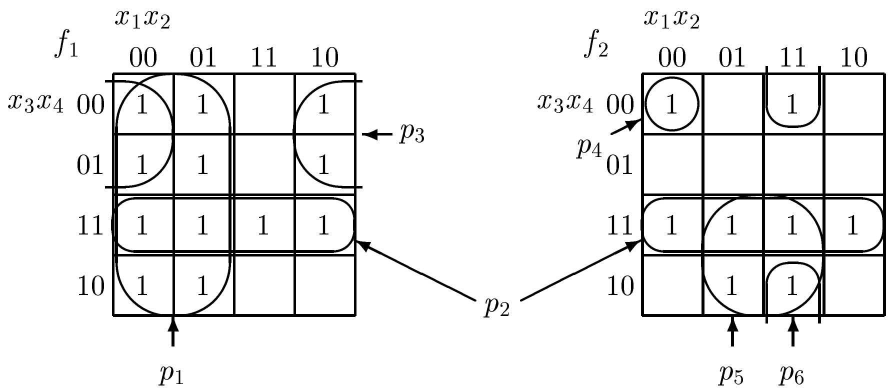
Problem 12.1: **PLA test.**

A missing crosspoint in the AND array at input line i will change the product term p_n, $p_n = i \cdot i_j i_k \ldots$, to p_n^f, $p_n^f = i_j i_k \ldots$. This corresponds to i stuck-at 1 in the product term (but not i stuck-at 1 in general). Thus, a missing crosspoint fault in the AND array corresponds to a literal stuck-at 1 in the sum-of-product representation.

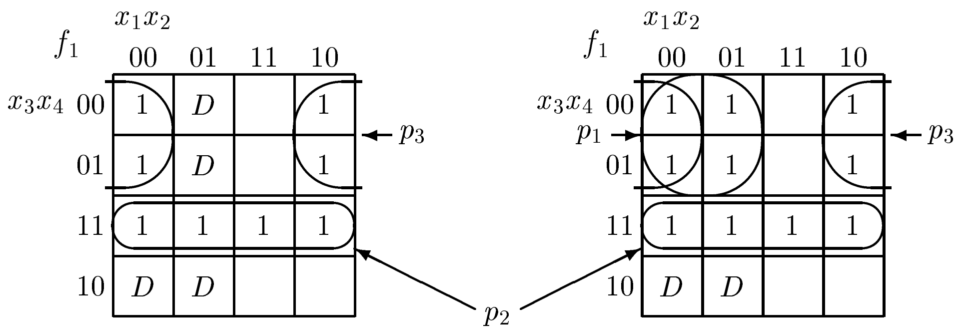
A missing crosspoint in the OR array at output line y will change the output y_n, $y_n = p \cdot p_q p_r \ldots$, to y_n^f, $y_n^f = p_q p_r \ldots$. This corresponds to p stuck-at 0 in the product term (but not p stuck-at 0 in general). Thus, a missing crosspoint fault in the OR array corresponds to a literal stuck-at 0 in the sum-of-product representation.

Problem 12.2: **PLA test.**

Question 1 The Karnaugh maps for the PLA are



Questions 2–3 A shrinkage fault (an extra transistor in the AND-plane) will reduce the corresponding minterm, p_1. p_1 only appears in the output f_1.



<u>Problem 12.2:</u> **continued.**

In the first case, p_1 completely disappears. The resulting Karnaugh map is shown in the above figure left.

In the second case, p_1 is diminished. The faulty minterm is $p_1^* = \overline{x_1}\ \overline{x_3}$.

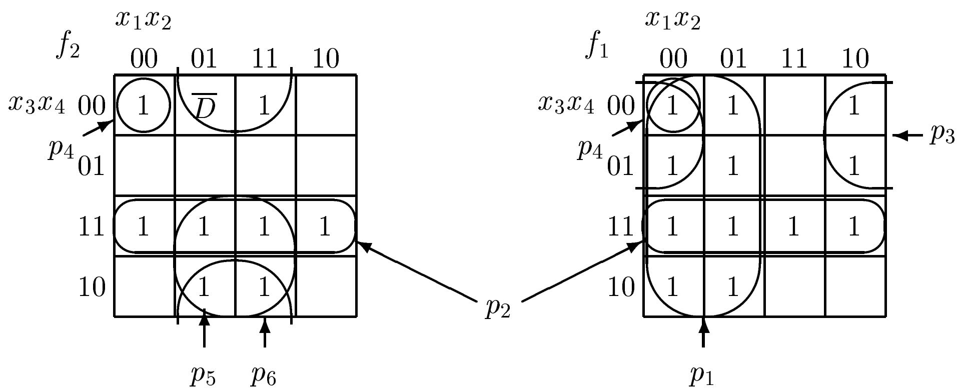
The resulting Karnaugh map is shown in the above figure right.

Questions 4–5 A growth fault (a missing transistor in the AND-plane) will enlarge the corresponding minterm, p_6, which only appears in the output f_2.

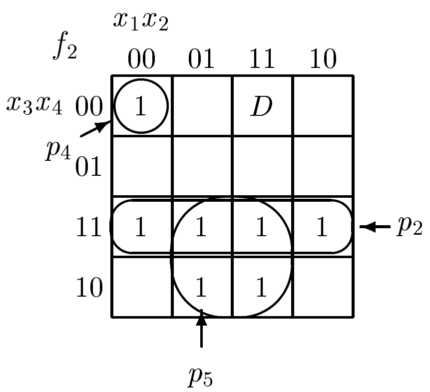
The faulty minterm is $p_6^* = x_2\overline{x_4}$. The resulting Karnaugh map is shown in the figure below left.

An appearance fault (an extra transistor in the OR-plane) introduces an extra minterm, p_4, in the output, f_1. However, in this case adding p_4 will not change f_1, and the redundant minterm will be very hard to detect.

The resulting Karnaugh map is shown in the figure below right.



Question 6 A disappearance fault (a missing transistor in the OR-plane) removes a minterm, p_6, from the output, f_2. The resulting Karnaugh map is shown in the figure below.



Problem 12.3: $\mathcal{C}$**-testability.**

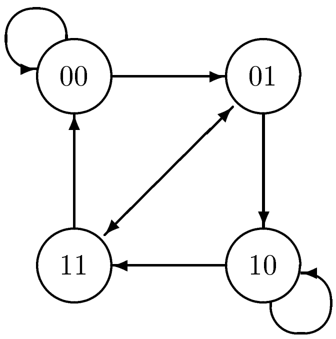
Question 1 The propagation D-cubes for the two-input EXOR gate are

$(D,0;D)$, $(D,1;\overline{D})$, $(\overline{D},0;\overline{D})$, and $(\overline{D},1;D)$.

$\oplus$	0	1	X	D	$\overline{D}$
0	0	1	X	D	$\overline{D}$
1	1	0	X	$\overline{D}$	D
X	X	X	X	X	X
D	D	$\overline{D}$	X	0	1
$\overline{D}$	$\overline{D}$	D	X	1	0

Thus, a single error will always be propagated to the output.

Question 2 4 patterns are required for an exhaustive test of the basic cell. All nodes are controllable and observable.
The constraint graph is shown below. Also, one set of test patterns, that exhaustively tests the array, is proposed.



	x_1	x_2	x_3	x_4	x_5	
t_1	0	0	0	0	0	$(0 \rightarrow 0)$
t_2	1	0	0	0	0	$(2 \rightarrow 2)$
t_3	0	1	1	1	1	$(1 \rightarrow 3)$
t_4	1	1	1	1	1	$(3 \rightarrow 1)$

Question 3 The results proven in Problem 2.2 do not hold for EXOR gates. Internal feedback invalidates the proof given. However, ignoring this fact for a moment, the argument proceeds as follows.

All primary inputs, x_i, are check points. Thus, a set of test patterns detecting all single stuck faults is given by two vectors ($2^n - 1$ other solutions may be found)

$$\{t_2, t_3\} \;=\; \{\; (0,1,1,1,\ldots)\;,\; (1,0,0,0,\ldots)\; \}$$

This result should come as no surprise. The cell-level exhaustive test set consists of four vectors. However, two vectors suffice to achieve full single stuck fault coverage. The fault coverage of the cell-level exhaustive test set is only slightly better, as a few double stuck faults may be detected (and possibly a number of other faults as well).

The test objective of $\mathcal{C}$-testability test sets is to exercise each basic cell through its entire truth table in order to detect any discrepancy. Thus, all stuck faults on the terminals of the basic cell are detected. In addition, many internal faults are detected as well, namely all faults that change the truth table of a single cell.

<u>Problem 12.4:</u> **$\mathcal{C}$-testability.**

Question 1 The iterative construction may multiply two binary, non-negative integers A and B. The multiplication is performed much like the usual paper-and-pen method, except for the row-wise (addition) carry propagation. The product P, $P = A \cdot B$, may be expressed as

$$A = \sum_{i=0}^{m} a_i \cdot 2^i \qquad B = \sum_{j=0}^{n} b_j \cdot 2^j$$

$$P = \sum_{i=0}^{m} a_i 2^i \cdot \sum_{j=0}^{n} b_j 2^j = \sum_{i=0}^{m} \sum_{j=0}^{n} (a_i b_j) 2^{i+j} = \sum_{k=0}^{m+n-1} p_k \cdot 2^k$$

An example may serve as an illustration of the process

$$\begin{array}{ccccccccccccc}
1 & 0 & 1 & 0 & 1 & 0 & \cdot & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
 & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & 1 & 1 & 0 & 0 & 1 & 1 & \\
 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & & \\
 & & & & 1 & 1 & 0 & 0 & 1 & 1 & & & \\
 & & & 0 & 0 & 0 & 0 & 0 & 0 & & & & \\
 & & 1 & 1 & 0 & 0 & 1 & 1 & & & & & \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0
\end{array}$$

The boundary conditions to be accomodated are that all c and d inputs at the top and all c inputs on the left are 0. Furthermore, all c and d outputs at the bottom row are fed into a ripple-carry adder. This structure has simple and well-defined properties and will not be considered in any detail at this point.

Question 2 Under the stated boundary conditions, controllability is severely limited. In an error-free array, a number of input combinations can not occur; For the top row, only combinations 0, 4, 8, and 12 can occur. For the second row, 0, 2, 4, 6, 8, 10, 12, and 14, can occur. For all subsequent rows, combinations 1, 3, 5, and 7 can not occur.

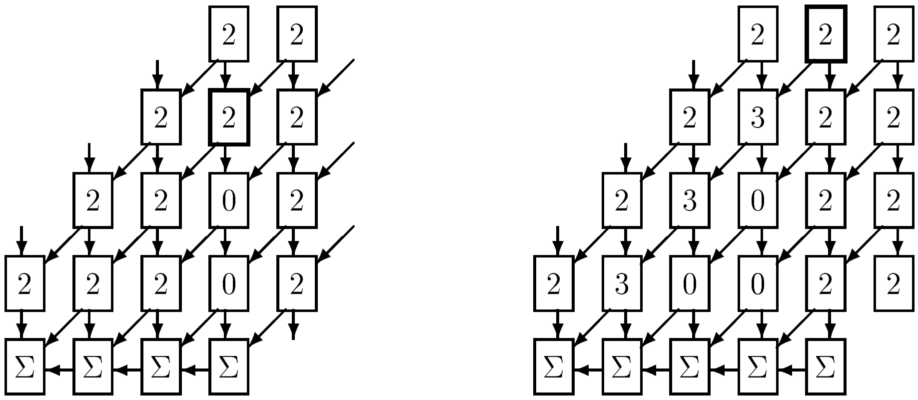
Thus, combinations 1, 3, 5, and 7 are not used in fault-free operation. The structure would still multiply correctly, if the truth table was changed for these input values.

The observability of the structure is reasonable. Specifically, the truth table shows that the s output is sensitive to any single change in inputs (and d only to some changes). If at most one cell-level fault is present at any time, fault effect propagation will take place through the s column until the fault is visible at the outputs. In some cases, the fault effect may be visible to the left below of the faulty cell.

Thus any single fault will always be visible at at least one output. Furthermore, most double faults will also be detected (this follows from considerations not discussed here).

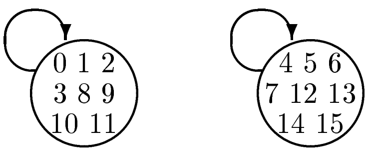
Problem 12.4: **continued.**

Question 3 Two fault propagation examples illustrate the properties of the array (fault-free situation all-twos).

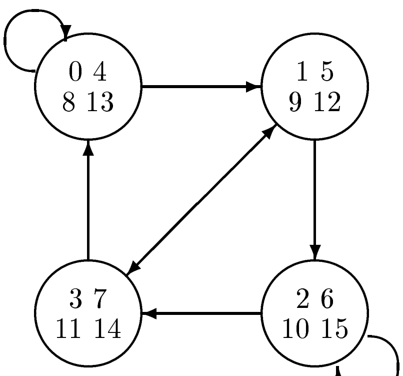


Question 4 Data flows between cells of the array across a row, down a column and along a diagonal. The 3 constraint graphs are shown below.

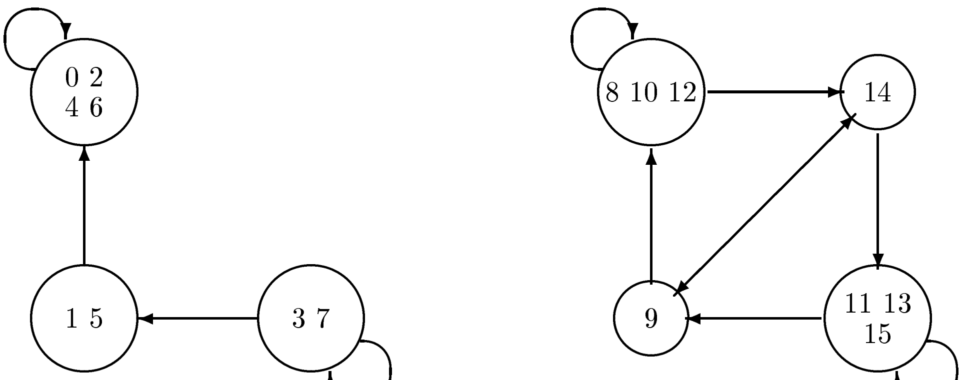
Horizontal constraints



Vertical constraints



Diagonal constraints



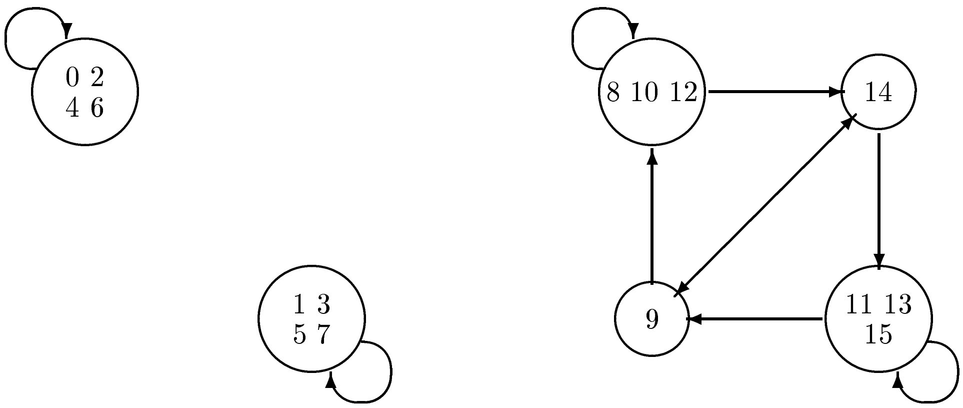
A necessary condition for $\mathcal{C}$-testability is, that all 16 input combinations may be applied to all cells in a cyclical manner. Thus, the c and d inputs of the top row must be fully controllable. However, it is easily seen that input combinations 1 and 5 can not be applied.

<u>Problem 12.4:</u> continued.

Question 5 A revised truth table is shown below. As s_o is not changed, fault propagation properties are not either.

graph vertex	a	b	s_i	c_i	s_o	c_o
0	0	0	0	0	0	0
1	0	0	0	1	1	**1**
2	0	0	1	0	1	0
3	0	0	1	1	0	1
4	0	1	0	0	0	0
5	0	1	0	1	1	**1**
6	0	1	1	0	1	0
7	0	1	1	1	0	1
8	1	0	0	0	0	0
9	1	0	0	1	1	0
10	1	0	1	0	1	0
11	1	0	1	1	0	1
12	1	1	0	0	1	0
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	1	1

Only the diagonal constraints are affected by the change. The revised diagonal constraint graph for the new structure is shown below.



The set of test patterns is determined by searching the constraint graphs. A minimal set consists of the following 16 patterns

Simple patterns :	0, 2, 4, 6, 10, 15
Pattern pairs :	$(1\ \ 3)$, $(5\ \ 7)$, $(9\ \ 14)^T$
Pattern blocks :	$\begin{pmatrix} 11 & 8 \\ 12 & 13 \end{pmatrix}$

The multiplier is now $\mathcal{C}$-testable and may be tested exhaustively no matter the array size using 16 test patterns.

Problem 12.4: **continued.**

Several design alterations have been made. These are

1) The functionality of the basic cell has been changed.
   The transistor count of the basic cell increases from 28 to 32.
2) All c and d inputs on the top row must be controllable.
   This may be achieved using a scan chain or a set of test controls.
3) All c inputs on the left must be controllable.
   This actual signal may be stolen from a neighbor using a test control that feeds an AND-gate.
4) The least significant byte of the result must be observable.
   This can be achieved using a scan path, if not a feature of the design.

The first two rows only allow a subset of values. This may be taken into advantage by realizing an irregular array with a specific cell for the first two rows. Hereby, the scan paths may be avoided using a set of test controls at the interface between rows 2 and 3.

8 Board-Level Testability

Problem 12.1: Interconnect test.

Question 1 The stuck-at tests require that all nets toggle. The two-net shorts tests require that the values of any pair of nets differ at least in a single pattern. Thus, if the sequence of test bits at any particular net is regarded as a code, the requirements are that any code is unique and that the all-1s and all-0s patterns are not used.

The solution outlined is sometimes referred to as the 'Counting' algorithm. The test sequence applied to each node is the binary representation of the nodes position. Proceeding in this manner, a total of $\lceil \log_2(k+2) \rceil$ test patterns results.

Question 2 $\lceil \log_2(2500+2) \rceil = 12$ patterns are required.

Question 3 The test response is a result of multiple faults. The multiple faults cause considerable confounding, as the test responses of several sets of faults are indistinguishable. The following fault hypotheses may result in the stated, faulty behaviour:

1. Nets 1 and 4 are wired-AND shorted, and net 3 is stuck-at-0,
2. nets 3 and 4 are wired-AND shorted, and net 1 is stuck-at-0,
3. nets 1, 3, and 4 are all wired-AND shorted,
4. nets 1, 3, and 4 are all stuck-at-0,

The pattern $(1, x, 1, 1)^T$ can be applied to reduce the diagnostic ambiguity:

$$\left.\begin{matrix} 1 \\ x \\ 1 \\ 1 \end{matrix}\right\} \longmapsto \begin{matrix} 1) \\ \left\{\begin{matrix} 1 \\ x \\ 0 \\ 1 \end{matrix}\right. \end{matrix} \quad \begin{matrix} 2) \\ \left\{\begin{matrix} 0 \\ x \\ 1 \\ 1 \end{matrix}\right. \end{matrix} \quad \begin{matrix} 3) \\ \left\{\begin{matrix} 1 \\ x \\ 1 \\ 1 \end{matrix}\right. \end{matrix} \quad \begin{matrix} 4) \\ \left\{\begin{matrix} 0 \\ x \\ 0 \\ 0 \end{matrix}\right. \end{matrix}$$

9 Fault-Tolerant Systems

<u>Problem 12.1:</u> Transparent RAM test.

The transformation of the March-C test sequence into a transparent version follows the recipe sketched in the following:

1. Initial March-C test sequence:

$$\Updownarrow (w0) \Uparrow (r0w1) \Uparrow (r1w0) \Downarrow (r0w1) \Downarrow (r1w0) \Updownarrow (r0)$$

2. Remove the initialization sequence:

$$\Uparrow (r0w1) \Uparrow (r1w0) \Downarrow (r0w1) \Downarrow (r1w0) \Updownarrow (r0)$$

3. Add a read operation in each string starting by a write:

$$\Uparrow (r0w1) \Uparrow (r1w0) \Downarrow (r0w1) \Downarrow (r1w0) \Updownarrow (r0)$$

4. Replace 0/1 by cell data $a_i/\overline{a_i}$:

$$\Uparrow (ra_i w\overline{a_i}) \Uparrow (r\overline{a_i} wa_i) \Downarrow (ra_i w\overline{a_i}) \Downarrow (r\overline{a_i} wa_i) \Updownarrow (ra_i)$$

5. If last sequence ends in $\overline{a_i}$, then add $(r\overline{a_i}wa_i)$:

$$\Uparrow (ra_i w\overline{a_i}) \Uparrow (r\overline{a_i} wa_i) \Downarrow (ra_i w\overline{a_i}) \Downarrow (r\overline{a_i} wa_i) \Updownarrow (ra_i)$$

The resulting transparent March-C test sequence is

$$\Uparrow (ra_i w\overline{a_i}) \Uparrow (r\overline{a_i} wa_i) \Downarrow (ra_i w\overline{a_i}) \Downarrow (r\overline{a_i} wa_i) \Updownarrow (ra_i)$$

10 Test Pattern Compaction

Problem 12.1: Test vector compatibility.

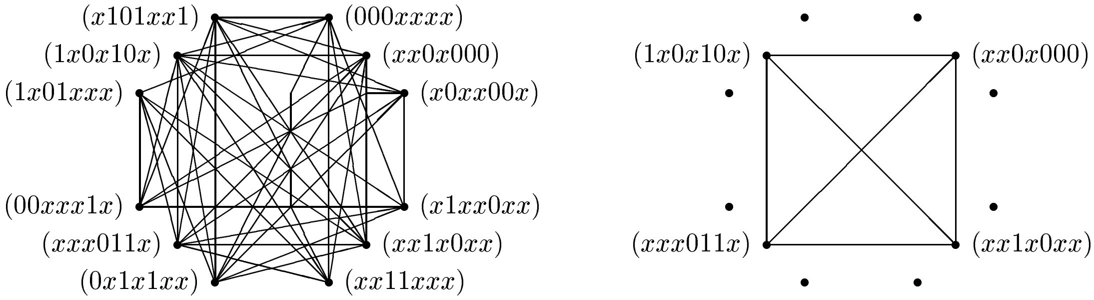
Assume that two compatible input vectors may be found, whose corresponding output vectors are not compatible. The situation is sketched below:

$$t_1 \;\; (1xxx0 \mid 1x0) \qquad\qquad t_2 \;\; (x1xx0 \mid x01)$$

Consider the last output bit. According to t_1, the value for the input pattern $\imath = (11xx0)$ is 0; According to t_2, the value is 1. As a consequence, t_1 and t_2 relates to different circuits, or the circuit is sequential.

Problem 12.2: Static test vector compaction.

The complete incompatibility graph as well as an excerpt of the graph is shown below for the 12 test patterns. It is obvious that compaction of the four clusters of test patterns will yield 4 compacted test patterns. The excerpt indicates that at least 4 vectors are required, i.e. the solution proposed is minimal.

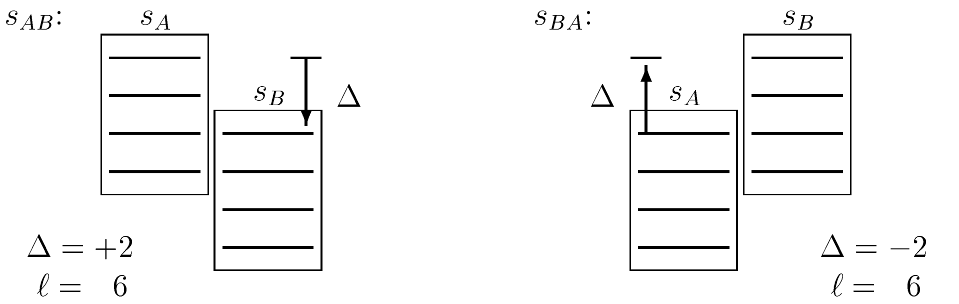


The proposed, compacted test patterns are

$$(1101101), \quad (000x000), \quad (001011x), \quad (x1110xx).$$

Problem 12.3: Static test set compaction.

Three test sequences are merged into a single sequence. In the analysis, the following notation is used to describe the properties of the merger of the test sequences s_A and s_B:



Problem 12.3: continued.

pattern	s_1	s_2	s_3
literals	24	24	30
don't cares	10	8	6
relative freedom	0.42	0.33	0.20

In the table above, the relative degree of freedom (the fraction of don't cares among the literals) is calculated for each sequence. First, an attempt is made to merge sequences s_1 and s_2. In the notation above, the shortest possible mergers are s_{12} (with $\Delta = +1$) and s_{21} (with $\Delta = -3$). The merged sequence s_{12} consisting of 5 test vectors is selected.

Next, an attempt is made to merge s_{12} and s_3. Similarly, the shortest possible mergers found are $s_{(12)3}$ (with $\Delta = +4$) and $s_{3(12)}$ (with $\Delta = -3$).

The shortest, merged sequence found is $s_{3(12)}$ consisting of 8 vectors:

$$\begin{array}{llll}
(001011) & s_3 & & \\
(0x0110) & | & & \\
(111xx0) & | & & \\
(000x10) & | & s_1 & \\
(000010) & | & | & s_2 \\
(11x101) & & | & | \\
(011001) & & | & | \\
(000x01) & & & |
\end{array}$$

Problem 12.4: Dynamic test compaction.

During dynamic compaction, the effect of a fault selected from the fault list is propagated to a yet unassigned output. If dynamic compaction fails, this implies that the D-chain ended without reaching an output. As usual, decisions are retried. If all decisions are exhausted and still no test is found, the fault may yet be testable.

First, fault effect propagation may have been blocked by assignments made before entering the dynamic compaction procedure. Thus, the fault may be testable through one of the unassigned inputs, but the test pattern is not compatible with the current assignments.

Second, the fault may be testable through (one of) the assigned outputs. A further investigation is necessary if the fault is to be dropped.

11 Automatic Test Equipment

12 Test Economy

<u>Problem 12.1:</u> 'Law of 10'.

The problem presented is simplified by the following assumptions:

1. For every IC, a *stochastic variable* X_i, $X_i \in \mathcal{B}(1, 1-Y)$, describes the presence or absence of a fault ($X_i = 1$ implies a faulty IC).
   All X_i's are assumed independent under both strategies.
2. Only one faulty IC is located in each PCB test.
3. The price of locating a faulty IC includes the cost of repair.
4. The cost of testing a fault-free PCB is negligible.

Under Strategy 1, the number of faulty ICs, k, we must test in order to find 100 good ICs, is determined by the *Pascal distribution* (or negative binomial distribution). The probability of finding 100 good ICs in $100 + k$ tests and the mean value are determined by

$$\begin{aligned} p\{Z\} &= \binom{k+100-1}{100-1}(1-Y)^k Y^{100} \\ E\{Z\} &= \frac{100(1-Y)}{Y} \end{aligned}$$

Under strategy 2, a PCB may contain several faulty ICs. Also, a faulty IC may be replaced by another faulty IC. A common pitfall is to replace only with tested ICs. The later situation is called Strategy 3.

Strategy 1 $100 + E\{Z\} = \frac{100}{Y}$ ICs must be tested. The cost is c_1

$$c_1 = \frac{100}{Y} \cdot (Ecu\ 1) = \underline{\underline{Ecu\ \frac{100}{Y}}}$$

Strategy 2 Repair is done using an untested IC (which may be faulty).
Thus, $E\{Z\}$ ICs are tested at total cost c_2

$$c_2 = \frac{100(1-Y)}{Y} \cdot (Ecu\ 10) = \underline{\underline{Ecu\ \frac{1000(1-Y)}{Y}}}$$

Strategy 3 Repair is done using a tested IC.
In the average, $100(1-Y)$ faulty ICs must be located and replaced by a tested IC, which, in turn, we will find for every $\frac{1}{Y}$ test.
The cost is c_3

$$c_3 = \underline{\underline{Ecu\ 100 \cdot (1-Y) \cdot (10 + \frac{1}{Y})}}$$

Problem 12.1: **continued.**

Comparison 1-2 Strategy 2 should be chosen over strategy 1 whenever

$$\begin{aligned} c_1 &\geq c_2 \\ \Longleftrightarrow Y &\geq 0.9 \end{aligned}$$

Thus, the simple replace strategy without incoming inspection wins at reasonable levels of yield.

Comparison 1-3 Strategy 3 should be chosen over strategy 1 whenever

$$\begin{aligned} c_1 &\geq c_3 \\ \Longleftrightarrow Y \cdot (Y - 0.9) &\geq 0 \\ \Longleftrightarrow Y &\geq 0.9 \end{aligned}$$

Comparison 2-3 Comparing strategies 2 and 3 ($c_2 \geq c_3$) yields

$$\begin{aligned} c_2 &\geq c_3 \\ \Longleftrightarrow \frac{10}{Y} &\geq 10 + \frac{1}{Y} \\ \Longleftrightarrow Y &\leq 0.9 \end{aligned}$$

Thus, as expected, strategy 2 is always preferable to strategy 3.

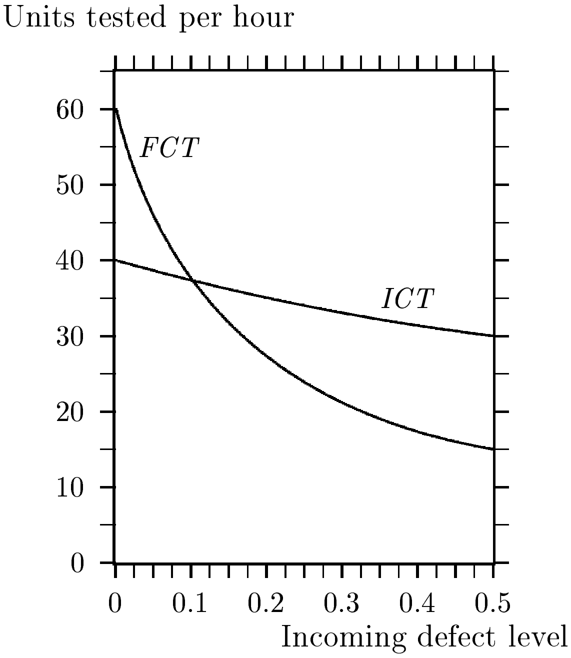
Problem 12.2: **Mixed test strategies.**

Question 1 The average test time is a function of production yield Y (or incoming defect level), as fault diagnosis is only performed on failing boards:

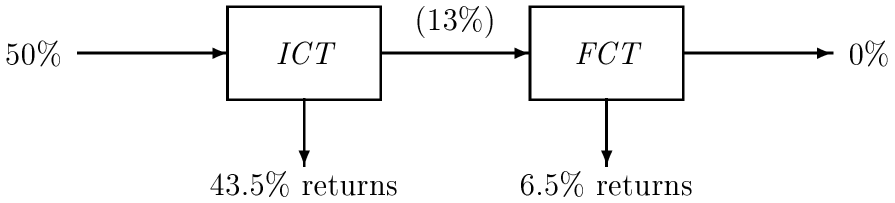
$$\begin{aligned} \text{FCT:} \qquad t &= \frac{1}{2} + \frac{1}{2} + (1 - Y) \cdot 6 \ \ \text{minutes} \\ \text{ICT:} \qquad t &= \frac{1}{2} + 1 + (1 - Y) \cdot 1 \ \ \text{minutes} \end{aligned}$$

From the expressions shown above, the number of units tested per hour is calculated. The result is shown in the figure below.

Problem 12.2: continued.



Question 2



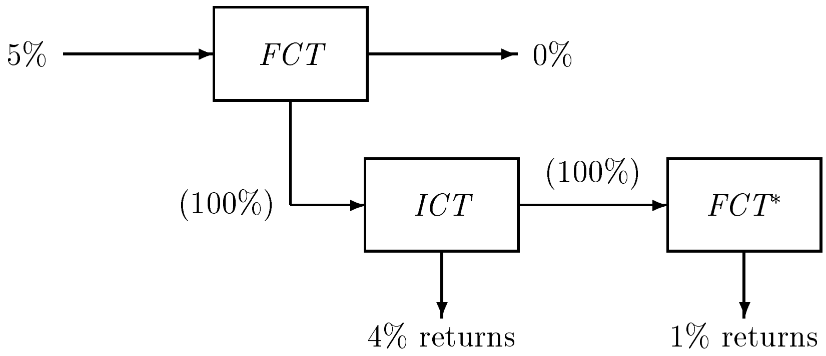
At 50% yield, the ICT tester is used to *prescreen* the incoming units at a rate of 30 units per hour. Part of the failures are captured at the ICT, the rest, 17 units per hour, are passed on to the FCT. For the FCT, the incoming yield is the defect level of the ICT:

$$\begin{aligned}\mathcal{DL} &= 1 - Y^{(1-\mathcal{FC})} \\ &= 1 - (\frac{1}{2})^{(1-0.8)} \quad = \quad 13\%\end{aligned}$$

At this defect level, the FCT can handle 34 units per hour, i.e. the FCT is idle half the time. The idle period is used to assist the ICT. In total, 37.5 units per hour can be processed. Using the FCT *alone* would result in 15 units being processed per hour.

Question 3 At higher levels of yield, prescreening does not remain effective. In the more complicated scheme outlined below, incoming boards are tested, not diagnosed, by the FCT. The ICT is used for diagnostics of failing boards. However, as the fault coverage is less than 100%, part of the failing boards are passed on to the FCT for diagnosis.

<u>Problem 12.2:</u> **continued.**



The FCT can test 60 units per hour. However, 1% of the tested units return for diagnosis. The cost of diagnosis is $6\frac{1}{2}$ times the cost of test alone, thus on the average 56 units are processed per hour. The unused capacity of the ICT is not considered.