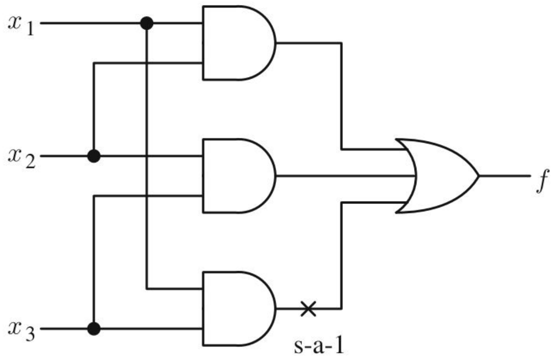
6.4. In Example 6.2 we have shown that the output function ofa 2-out-of-3 majority voter is self-dual. Figure 6.9 shows a logic circuit implementing a 2-out-of-3 majority voter. Find an input assignment which detects the stuck-at-1 fault marked by a cross.



x_1	x_2	x_3		f	f^α
0	0	0		0	1
0	0	1		0	1
0	1	0		0	1
0	1	1		1	1
1	0	0		0	1
1	0	1		1	1
1	1	0		1	1
1	1	1		1	1

Inputs to detect the s-a-1 fault: {0,0,0},{0,0,1},{0,1,0},{1,0,0} → Any input with none or one 1's

<u>6.8.</u> Recomputing with shifted operands is used to protect a k-bit ripple-carry adder from permanent faults. A 2-bit left shift is used. Suppose that the bits slices i and $i + k/2 + 1$ have a permanent fault, for $i \in \{0, 1, ..., k/2 - 1\}$. Will such a fault be detected for any A, B, and k? Illustrate your answer in the example of some operands A and B and k = 8.

Yes, these faults will be detected for any A,B and k.

The permanent faults will be in i and $i + 5$. For this example, I assume $i = 1$ and both bits are stuck at 0.

- ❏ At time t_0

A			0	1	0	1	0	1	0
B			0	0	0	1	1	1	1
R			0	1	1	1	0	0	1

- ❏ At time t_1

A	0	1	0	1	0	1	0		
B	0	0	0	1	1	1	1		
R	0	1	0	1	0	0	1		

0111001 $\neq$ 0101001 $\rightarrow$ Faults detected !

Because the two fault are separated by $k/2 + 1$ bits, recomputing with shifted operands has detected the faults.

<u>6.10.</u> Repeat problem 6.8 for recomputing with swapped operands.

- At time t_0

A	0	0	1	0	0	0	1	0
B	0	1	0	0	0	1	1	1
R	0	1	1	0	1	0	0	1

- At time t_1

A	0	0	1	0	0	0	1	0
B	0	1	1	1	0	1	0	0
R	1	0	0	1	0	1	1	0

0111001 = 0111001 → Faults not detected, because swapping the inputs has no effect if the fault is a s-a-0 or s-a-1 on the output.

<u>6.16</u>. Recomputing using duplication with comparison is applied to perform the bit-wise XOR on k-bit operands A and B, where k is even. Suppose that the bit slices 0 and k/2 are permanently stuck to 1. Will such a fault be detected for any A, B, and k? Illustrate your answer in the example of A =[10100111], B =[01101110] and k = 8.
In this case, the faults can not be detected if some bits of the output result are stuck at an specific value, because when comparing both results, those bits will have always the same value. This can be seen in the following example:

First, the bit-wise XOR is performed in duplicated lower halves of the operands:

A	0	1	1	1	0	1	1	1
B	1	1	1	0	1	1	1	0
XOR	1	0	0	1	1	0	0	1

Then, the bit-wise XOR is performed in duplicated higher halves of the operands:

A	1	0	1	0	1	0	1	0
B	0	1	1	0	0	1	1	0
XOR	1	1	0	1	1	1	0	1

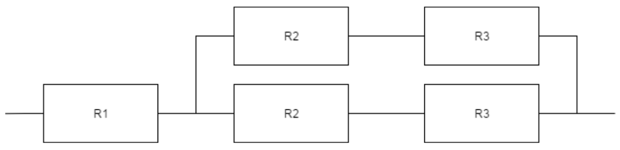
Both results are equal, but the results are not correct.

<u>7.6</u> An engineer designs a software system consisting of three modules in series with the reliabilities R1 = 0.89, R2 = 0.85, and R2 = 0.83 per 6 × e04 s of CPU execution time. It is acceptable to add two redundant modules to the system (diverse versions). Which of the following is best to do:

1. **Duplicate modules 2 and 3 using high-level redundancy.**
2. **Duplicate modules 2 and 3 using low-level redundancy.**
3. **Triplicate the module 3.**

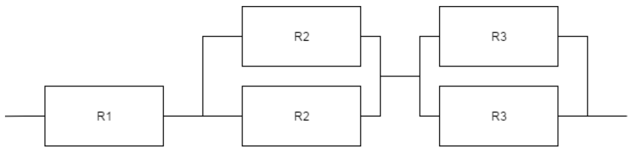
Analysing all three options:

1.



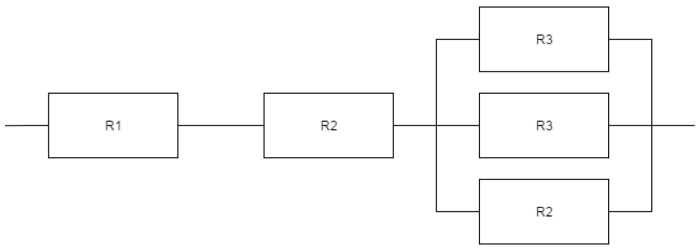
$$R_{total} = R_1 \cdot (1 - (1 - R_2 \cdot R_3)^2) = 0.8128$$

2.



$$R_{total} = R_1 \cdot (1 - (1 - R_2)^2) \cdot (1 - (1 - R_3)^2) = 0.8448$$

3.



$$R_{total} = R_1 \cdot R_2 \cdot (1 - (1 - R_3)^3) = 0.7528$$

The best option is to duplicate modules 2 and 3 using low-level redundancy.

<u>6)</u> Check if the following functions are self-dual:

A. f(a,b,c) = a’c’ + b’c + a’b

$$\overline{f(\overline{a},\overline{b},\overline{c})} = \overline{ac+b\overline{c}+a\overline{b}} = \overline{ac}\cdot\overline{b\overline{c}}\cdot\overline{a\overline{b}} = (\overline{a}+\overline{c})\cdot(\overline{b}+c)\cdot(\overline{a}+b) =$$
$$(\overline{a}\overline{b}+\overline{a}\overline{c}+\overline{c}\overline{b})\cdot(\overline{a}+b) = \overline{a}\overline{a}\overline{b}+\overline{a}\overline{a}\overline{c}+\overline{a}\overline{b}\overline{c}+\overline{a}\overline{b}b+\overline{a}\overline{c}b+\overline{c}\overline{b}b = \overline{b}+\overline{c}+\overline{a}\overline{b}\overline{c}+\overline{a}\overline{c}b$$

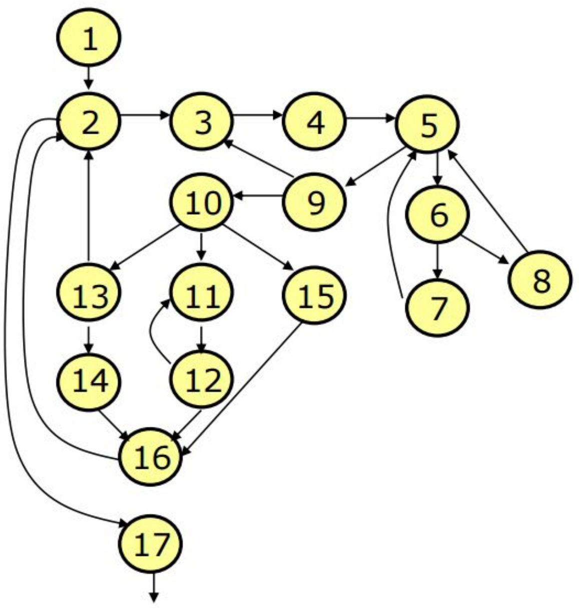
No self-dual

B. f(a,b,c) = b + a’c’

$$\overline{f(\bar{a},\bar{b},\bar{c})} = \overline{\bar{\bar{b}}+ac} = b\cdot(\overline{ac}) = b\cdot(\bar{a}+\bar{c})$$

No self-dual

<u>7)</u> Compute pre- and post-dominator trees for vertices of the flowgraph shown below. Find the kernel sets.



- Pre-dominator tree:

$$\begin{tikzcd}
 & & & & 1 \arrow[d] & & \\
 & & 17 & & 2 \arrow[ll] \arrow[d] & & \\
 & & & & 3 \arrow[d] & & \\
 & & & & 4 \arrow[d] & & \\
 & & & & 5 \arrow[dl] \arrow[dr] & & \\
 & & & 9 \arrow[d] & & 6 \arrow[d] \arrow[dr] & \\
 & & & 10 \arrow[dlll] \arrow[dll] \arrow[dr] \arrow[drr] & & 7 & 8 \\
16 & 11 \arrow[d] & & & 13 \arrow[d] & 15 & \\
 & 12 & & & 14 & &
\end{tikzcd}$$

- Post-dominator tree:

$$\begin{tikzcd}
 & & 17 \arrow[d] & & & & & & & \\
 & & 2 \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drrrrr] & & & & & & & \\
1 & 13 & & 16 \arrow[dl] \arrow[d] \arrow[drr] & & & & 10 \arrow[d] & & \\
 & & 14 & 12 \arrow[d] & & 15 & & 9 \arrow[d] & & \\
 & & & 11 & & & & 5 \arrow[dlll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
 & & & & 4 \arrow[d] & & 7 & & 8 & 6 \\
 & & & & 3 & & & & &
\end{tikzcd}$$

❑ Kernel

$$L_{pre} = \{7, 8, 12, 14, 15, 16, 17\}$$

$$L_{post} = \{1, 3, 6, 7, 8, 11, 13, 14, 15\}$$

$$L_{pre}^{D} = \{16, 17\}$$

$$L_{post}^{D} = \{1, 3, 6, 13\}$$

$$L_{pre} - L_{pre}^{D} = \{7, 8, 12, 14, 15\}$$

$$L_{post} - L_{post}^{D} = \{7, 8, 11, 14, 15\}$$

$$Kernel\ = \{7, 8, 11, 14, 15\}\ or\ \{7, 8, 12, 14, 15\}$$