

- repair team dependency: if several blocks are repaired by the same repair team, then a failing block has to wait to be repaired if the repair team is busy with another failed block. This constitutes a functional dependency which can be modelled as illustrated in Figure 48;
- collective repair: several blocks are repaired within the same repair operation;
- standby redundancy: when the active block fails, this starts the standby block (see Figure 11);
- spare parts: when an active block fails, the repair may need to use some spare parts. Therefore, the repair is possible only if one spare part is available. In addition, the spare part used to repair one block becomes unavailable to repair another block;
- blocks in series: when one of the blocks in series fails or is under repair, the others may be stopped (e.g. because the output of the failed block is needed for later blocks in the series).
- etc.
- Events able to occur only in a given order (an event cannot occur before another one has occurred):
 - the repair of a block cannot start before the block has failed. Such functional dependency has been already handled with ordinary RBDs for availability, reliability and frequency calculations;
 - for a given set of blocks (B_1, B_2, \dots, B_n) , the repair starts only when all of them have failed;
 - the blocks become non-repairable after the whole system has failed. This is what happens when reliability calculations are performed;
 - more generally, for a set of events (e_1, e_2, \dots, e_n) , this implies that e_2 cannot occur as long as e_1 has not occurred, that e_3 cannot occur as long as e_2 has not occurred, ..., that e_n cannot occur as long as e_{n-1} has not occurred. In other words, e_1 is inhibited by \bar{e}_2 , e_2 is inhibited by \bar{e}_3 , ..., e_n is inhibited by \bar{e}_{n-1} . Therefore, the events can only occur in the sequence e_1, e_2, \dots, e_n . This may be the case when an electrical device cannot be started before the electrical power is switched on or when a cold standby device cannot be activated before the failure of the active device. This interaction is similar to sequential gates (often noted SEQ) found in dynamic fault tree analysis (see the SEQ gate in Table 4);
 - etc.

12.2.3 Systemic dynamic interactions

Those interactions do not necessarily imply functional dependencies between the blocks which may behave independently from each other. They occur when the ordinary logical rules cannot be used.

Examples are the following:

- m/n majority vote: this logical configuration has already been analysed (see 7.5.1 and 9.4) and a special logical gate has been introduced to model it.
- Events which shall occur in a given order:
 - demand triggering an action performed by a given block B: if the demand occurs before B has failed, the action is performed and the system remains in up state, if the demand occurs after B has failed, the action is not performed and the system fails;
 - isolation valve protecting a system against overpressure: a hazardous event occurs only if the valve is opened before the pressure has been dropped down upstream the isolation valve;
 - more generally, for a set of events e_1, e_2, \dots, e_n , the output is produced only if the events occur in this given order, otherwise no output is produced. This interaction is similar to the "priority" AND gates (often noted PAND) found in dynamic fault tree

analysis and which can also be used for DRBDs. This may be represented as a gate combining the input of several blocks.

Special gates are needed to represent the systemic dynamic dependencies as, for example, the m/n and the PAND or SEQ gates presented in Table 4 and which are popular extensions of dynamic fault trees.

The m/n gate has already been analysed and PAND and SEQ gates are analysed hereafter (see Figure 49 to Figure 52). The symbols usually implemented in dynamic fault trees have been used here but NOT gates have been inserted in inputs and outputs in order to keep the coherence with regards to the RBD logic.

12.2.4 Graphical representations of dynamic interactions

As said in 12.2.2 and 12.2.3, the kinds of possible dynamic interactions are virtually endless. Therefore, even if some attempts have been made (see references [15], [16] and [17]) to propose graphical symbols for specific cases, this does not cover all the cases and only some basic graphical elements can be proposed in this standard.

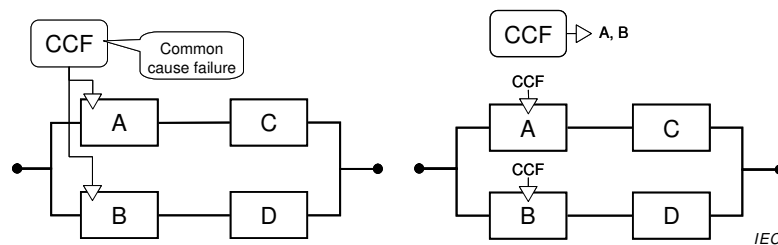


Figure 46 – Dynamic interaction between a CCF and RBDs' blocks

Figure 46 shows the strong interactions (i.e. strong functional dependencies) between an external element and some blocks: blocks A and B fail when the common cause failure represented by the external block CCF occurs.

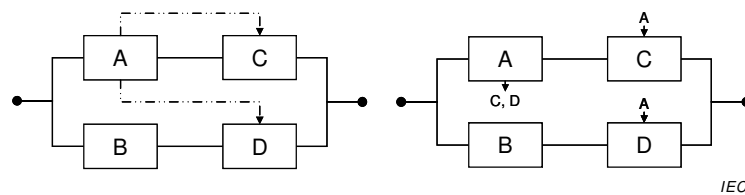


Figure 47 – Various ways to indicate dynamic interaction between blocks

Figure 47 shows two ways to represent the interaction (i.e. functional dependencies) between blocks: the state of blocks C and D depends on the state of block A.

The same mechanisms have been implemented in Figure 48 to represent the interaction between the single repair team and the repaired blocks.

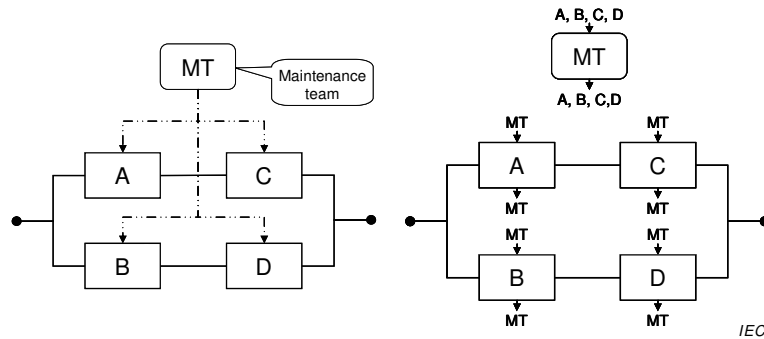


Figure 48 – Dynamic interaction between a single repair team and RBDs' blocks

These simple graphical representations aim only at indicating that there is some dynamic interaction between the blocks and the external elements. The dotted lines on the left hand side of Figure 47 and Figure 48 can be used when only few interactions have to be represented in an RBD. When there are many interactions to be represented, the proposal on the right hand side of Figure 47, Figure 46 and Figure 48 is clearer. The very nature of the interactions themselves should be specified elsewhere. The main use of these representations is to support the graphical presentation of the RBD and to ensure that the external elements are well identified.

Figure 49 shows how a PAND gate can be used within a DRBD: the output O goes to the down state only if I_1 goes to the down state before I_2 goes to the down state.

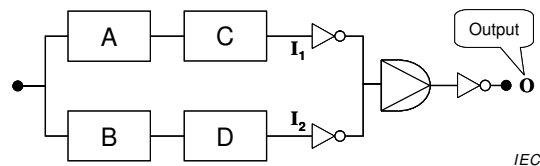


Figure 49 – Implementation of a PAND gate

The functioning of the PAND gate is illustrated in Figure 50. The PAND gate is equivalent to the 5 states of the finite-state automaton drawn on the left hand side of the figure.

- State 1: I_1 and I_2 are in up state. Then the output O is in the up state.
- State 2: I_2 has gone to the down state first and I_1 is still in the up state. Then the output O is in the up state.
- State 3: I_1 has gone to the down state first and I_2 is still in the up state. Then the output O is in the up state.
- State 4: I_1 and I_2 have gone to the down state but I_2 has gone first. Then the output O is in the up state.
- State 5: I_1 and I_2 have gone to the down state but I_1 has gone first. Then the output O has gone to the down state.

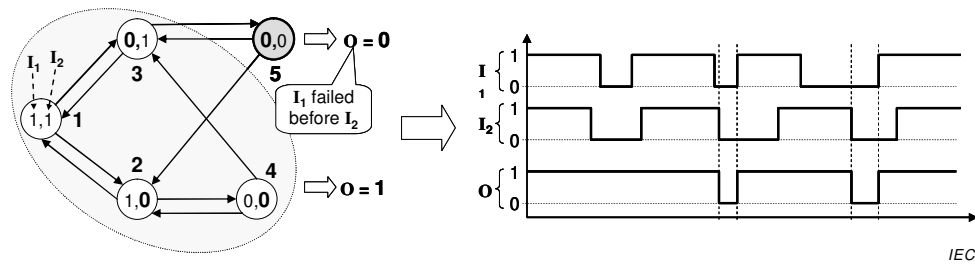


Figure 50 – Equivalent finite-state automaton and example of chronogram for a PAND gate

Then, when the input I_1 and I_2 varies between 1 and 0, the output of the PAND gate (Figure 49) changes according to the rules presented by this finite-state automaton. This gives, for example, the chronogram presented on the right hand side of Figure 50. A Petri net modelling the same finite-state automaton is analysed in Annex E and Figure E.6.

Figure 51 shows how a SEQ gate can be used within a DRBD: as for the PAND gate, the output O goes to the down state only if I_1 goes to the down state before I_2 goes to the up state. The difference is that I_2 cannot go to the down state before I_1 has gone to the down state first. Therefore, the failure of B and D are inhibited as long as I_1 is in up state and this is indicated thanks to the dynamic interactions drawn in dotted lines.

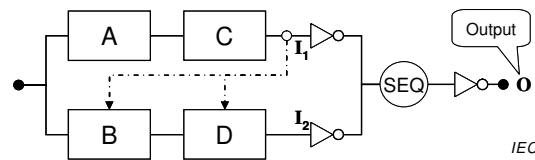


Figure 51 – Implementation of a SEQ gate

The functioning is illustrated in Figure 52. The SEQ gate is equivalent to the 5 states finite-state automaton drawn on the left hand side of the figure. The states are the same as for the PAND gate except that there is no transition from state 1 to state 2 in order to force the order of the failures: I_1 first, then I_2 .

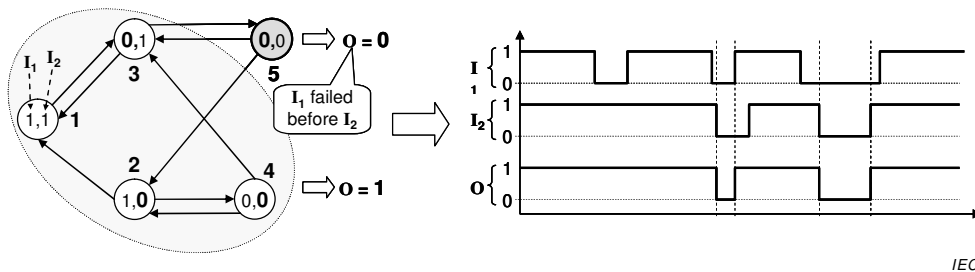


Figure 52 – Equivalent finite-state automaton and example of chronogram for a SEQ gate

As shown in the chronogram, I_2 cannot fail before I_1 has previously failed.

A Petri net modelling the same finite-state automaton is analysed in Annex E, Figure E.6 and Figure E.7.

12.2.5 Probabilistic calculations

Making the probabilistic calculation by using the Markovian approach is proposed in literature (see references [2], [29] and [30]). Nevertheless, building a Markov process for a whole DRBD is quickly limited by the combinatorial explosion of the number of states. Therefore, this approach should be restricted to small independent parts of the DRBD as this has been done for the RBD driven Markov processes described in Clause C.4.

Another approach which is proposed in literature is to make the link between DRBDs and finite state automata (state-events machine or Petri net). This is more effective than the markovian approach but the analytical calculations are no longer possible and Monte Carlo simulation has to be implemented.

The RBD driven Petri nets described in Annex E are an effective way to mix the RBD and PN approaches in order to deal with dynamic RBDs problems and calculations.

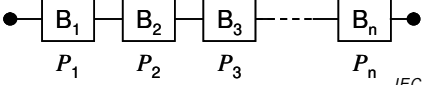
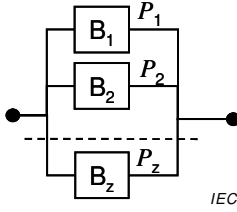
Annex A (informative)

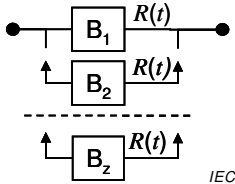
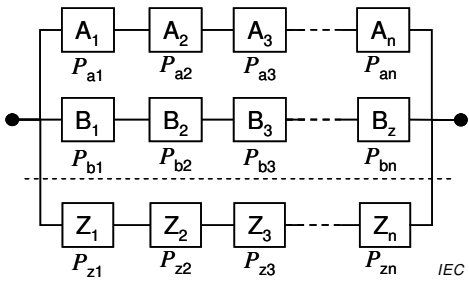
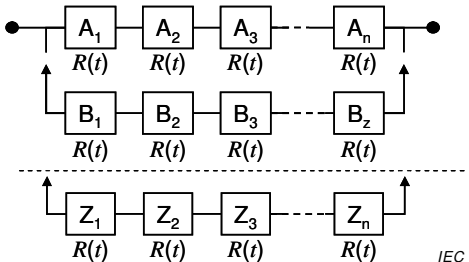
Summary of formulae

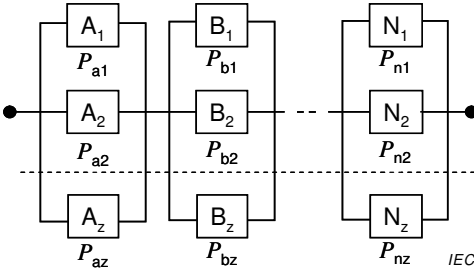
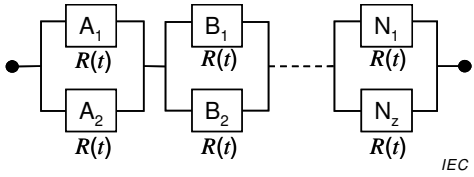
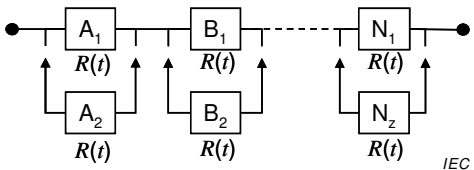
Warning: The formulae presented in Table A.1 are intended to be used by users aware of the underlying hypothesis and mathematics and of the limitations when approximations are implemented.

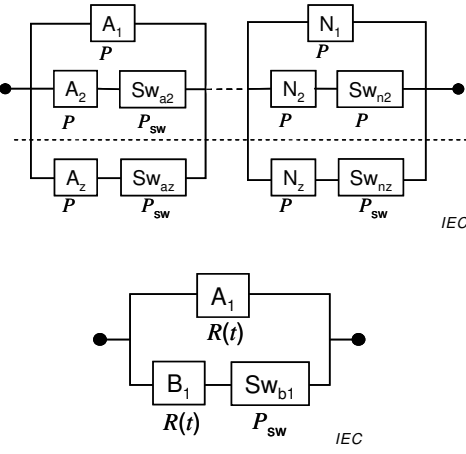
NOTE In Table A.1, frequent use is made of the terms “active” and “standby”. The former is used to indicate that the blocks concerned (each of which can consist of a component, sub-system, system, etc.) are energized (powered-up) and hence are liable to failure. The latter on the other hand is used to indicate that the block or blocks concerned are de-energized (powered-down) and not liable to failure.

Table A.1 – Example of equations for calculating the probability of success of basic configurations

Basic configuration	Equation for system $P_S, R_S(t), A_S(t)$
1 Series structures 	<p>A General case Constant probabilities: $P_S = P_1 \cdot P_2 \dots P_n$ Time dependent probabilities: $R_S(t) = R_1(t) \cdot R_2(t) \dots R_n(t)$ $A_S(t) = A_1(t) \cdot A_2(t) \dots A_n(t)$</p> <p>B With $P_1 = P_2 = \dots P_n = P$ $\Rightarrow P_S = P^n$</p> <p>C With $R_1(t) = R_2(t) = \dots R_n(t) = R(t)$ $\Rightarrow R_S(t) = R(t)^n$</p> <p>D With $A_1(t) = A_2(t) = \dots A_n(t) = A(t)$ $\Rightarrow A_S(t) = A(t)^n$</p>
2 Parallel structures Active 	<p>A Active general case Constant probabilities: $P_S = 1 - (1 - P_1) \cdot (1 - P_2) \dots (1 - P_z)$ Time dependent probabilities: $R_S(t) : \text{no simple general formula (see NOTE 1)}$ $A_S(t) = 1 - [1 - A_1(t)] \cdot [1 - A_2(t)] \dots [1 - A_z(t)]$</p> <p>B With $P_1 = P_2 \dots = P \Rightarrow P_S = 1 - (1 - P)^z$</p> <p>C With $A_1(t) = A_2(t) = \dots A_z(t) = A(t)$ $\Rightarrow A_S(t) = [1 - A(t)]^z$</p>

Basic configuration	Equation for system $P_S, R_S(t), A_S(t)$
<p>Standby</p> 	<p>D Standby with $R(t) = A(t) = e^{-\lambda \cdot t}$ (i.e., non-repaired items)</p> $R_S(t) = A_S(t) = e^{-\lambda \cdot t} + \lambda \cdot t \cdot e^{-\lambda \cdot t} + \dots + \frac{(\lambda \cdot t)^{z-1} e^{-\lambda \cdot t}}{(z-1)!}$
<p>3 Series-parallel structures (redundant systems)</p> <p>Active</p> 	<p>A Active general case</p> <p>Constant probabilities</p> $P_S = 1 - \prod_{i=a}^z (1 - P_{i1} \cdot P_{i2} \dots P_{in}) = 1 - \prod_{i=a}^z [1 - \prod_{j=1}^n P_{ij}]$ <p>Time dependent probabilities:</p> <p>$R_S(t)$: no simple general formula (see NOTE 1)</p> $A_S(t) = 1 - \prod_{i=a}^z [1 - \prod_{j=1}^n A_{ij}(t)]$ <p>B Active with</p> $P_{i1} = P_{i2} = \dots = P_i \quad \forall i \Rightarrow P_S = 1 - \prod_{i=a}^z (1 - P_i^n)$ <p>C Active with</p> $A_{i1}(t) = A_{i2}(t) = \dots = A_i(t) \quad \forall i \Rightarrow A_S(t) = 1 - \prod_{i=a}^z [1 - A_i(t)^n]$ <p>D Active with</p> $P_{ij} = P \quad \forall i, j \Rightarrow P_S = 1 - \prod_{i=a}^z (1 - P^n)$ <p>E Active with</p> $A_{ij}(t) = A(t) \quad \forall i, j \Rightarrow A_S(t) = 1 - [1 - A(t)^n]^z$
<p>Standby</p> 	<p>F Standby with $R(t) = A(t) = e^{-\lambda t}$ (i.e., non-repaired items)</p> $R_S(t) = A_S(t) = e^{-n\lambda t} + n\lambda t \cdot e^{-n\lambda t} + \dots + \frac{(n\lambda t)^{z-1} e^{-n\lambda t}}{(z-1)!}$

Basic configuration	Equation for system P_S , $R_S(t)$, $A_S(t)$
4 Parallel-series structures (redundant elements)	
<p>Active</p>  	<p>A Active general case Constant probabilities</p> $P_S = \prod_{i=a}^n \left\{ 1 - \prod_{j=1}^z (1 - P_{ij}) \right\}$ <p>Time dependent probabilities: $R_S(t)$: no simple general formula (see NOTE 1)</p> $A_S(t) = \prod_{i=a}^n \left\{ 1 - \prod_{j=1}^z [1 - A_{ij}(t)] \right\}$ <p>B Active with $P_{i1} = P_{i2} = \dots = P_i \quad \forall i \quad \Rightarrow P_S = \prod_{i=a}^n [1 - (1 - P_i)^z]$</p> <p>E Active with $A_{i1}(t) = A_{i2}(t) = \dots = A_i(t) \quad \forall i$ $\Rightarrow A_S(t) = \prod_{i=a}^n \{1 - [1 - A_i(t)]^z\}$</p> <p>F Active with $P_{ij} = P \quad \forall i, j \quad \Rightarrow P_S = [1 - (1 - P)^z]^n$</p> <p>G Active with $A_{ij}(t) = A(t) \quad \forall i, j \quad \Rightarrow A_S(t) = \{1 - [1 - A(t)]^z\}^n$</p> <p>H Assuming $R(t) = A(t) = e^{-\lambda \cdot t}$ $R_S(t) = A_S(t) = (2 \cdot e^{-\lambda \cdot t} - e^{-2 \cdot \lambda \cdot t})^n$</p>
<p>Standby</p> 	<p>D Standby with $R(t) = A(t) = e^{-\lambda \cdot t}$ $R_S(t) = A_S(t) = (e^{-\lambda \cdot t} + \lambda \cdot t \cdot e^{-\lambda \cdot t})^n$</p>

Basic configuration	Equation for system $P_S, R_S(t), A_S(t)$
<p>5 Parallel-series structures (redundant elements)</p> 	<p>A Active with $P_{ij} = P \quad \forall i, j$ except P_{sw}</p> $\Rightarrow P_S = [1 - (1 - P) \cdot (1 - P \cdot P_{sw})^{z-1}]^n$ <p>B Active with $A_{ij}(t) = A(t) \quad \forall i, j$ except $A_{sw}(t)$</p> $\Rightarrow A_S(t) = \left\{ 1 - [1 - A_S(t)] \cdot [1 - A_S(t) \cdot A_{sw}(t)]^{z-1} \right\}^n$ <p>C Active with $z = 2, n = 1$</p> <p>and $R_{ij}(t) = A_{ij}(t) = e^{-\lambda t} \quad \forall i, j$ except P_{sw}</p> $\Rightarrow R_S(t) = A_S(t) = e^{-\lambda t} + P_{sw}e^{-\lambda t} - P_{sw}e^{-2\lambda t}$
<p>NOTE 1 In case of non-repaired blocks $R_S(t) = A_S(t)$.</p> <p>NOTE 2 For non-repaired blocks with constant failure rates, P can be replaced by $R(t) = A(t) = e^{-\lambda t}$.</p> <p>NOTE 3 Formulae for standby systems are based on the assumption that the reliability of switching and sensing mechanisms is 100 % ($P_{sw} = 1$).</p>	

Annex B (informative)

Boolean algebra methods

B.1 Introductory remarks

Apart from the use of Boolean truth tables (see 11.4) and binary decision diagrams (see 11.5), the analysis of RBDs as described so far makes use mainly of conventional algebraic mathematical formulae. However, Boolean algebra in general can also be used for such analyses, and in many instances is much more efficacious and straightforward. In particular, the use of Boolean algebra may well be the most straightforward approach whenever

- a) RBDs contain repeated blocks (see Figure 37),
- b) RBDs contain directional arrows (see Figure 10 and Figure 35),
- c) the system is particularly complicated,
- d) it is easier to construct a Boolean expression for system success (or failure) than it is to construct an RBD,
- e) the system comprises a number of blocks too large to be tractable by simple formulae.

Item d) of the above list is worthy of note. For many systems and networks the listing of equipment success (or failure) combinations in Boolean terms is often a more straightforward task than the construction of the corresponding RBD. By employing at the outset the Boolean approach to analyse the system, the risk of making errors in the course of constructing the RBD is entirely avoided.

Item e) of the above list may be related to RBDs modelling industrial systems with a dozen of components and leading to the combinatorial explosion of the terms to be taken into account in the formulae. This is particularly crucial when numerous repeated blocks also have to be managed.

B.2 Notation

The conventional symbols \cup and \cap denoting the logical "OR" and "AND" play for the Boolean algebra the same role as the addition (+) and of the multiplication (\cdot) for ordinary algebra. This is why, in what follows, it has been found more convenient, to use a "+" symbol to denote logical "OR" and a full stop "." to denote logical "AND"². As usual a bar over a Boolean variable will denote the inverse or complement of the variable concerned: e.g. \bar{a} is interpreted as "not a ". For example $a \cdot b \cdot \bar{c} \cdot e + f \cdot g$ is to be interpreted " a AND b AND NOT c AND e OR f AND g ". The context in which the symbols are used should make the meaning clear.

² The advantage of such a notation becomes apparent in Annex B where expressions of the type $S = a \cdot b + \bar{a} \cdot e \cdot b + \bar{a} \cdot e \cdot d + a \cdot \bar{b} \cdot e \cdot d + \bar{a} \cdot c \cdot d + a \cdot \bar{b} \cdot c \cdot d$ are frequently found. Taking this latter expression as an example and writing it using set theory symbols, one obtains: $S = a \cap b \cup \bar{a} \cap e \cap b \cup \bar{a} \cap e \cap d \cup a \cap \bar{b} \cap e \cap d \cup \bar{a} \cap c \cap d \cup a \cap \bar{b} \cap c \cap d$ which for many readers may be quite difficult to interpret or evaluate.

B.3 Tie sets (success paths) and cut sets (failure paths) analysis

B.3.1 Notion of cut and tie sets

As said in 8.1, an RBD can be considered as an electrical circuit (see Figure 14) and this analogy is useful to identify:

- the tie sets which correspond to a closed electrical circuit and represent the combinations of the blocks in up states leading to the system being in the up state. The tie sets are also the "success" paths of the RBD;
- the cut sets which correspond to a cut electrical circuit and represent the combinations of the blocks in down states leading to the system being in the down state. The cut sets are also the "failure" paths of the RBD.

Using this analogy allows to transform the RBD presented in Figure 10 into the electrical circuit presented in Figure 15. From this representation it is easy to identify various tie sets of this RBD and Figure B.1 and Figure B.2 show various examples of combinations of closed switches corresponding to the system up state.

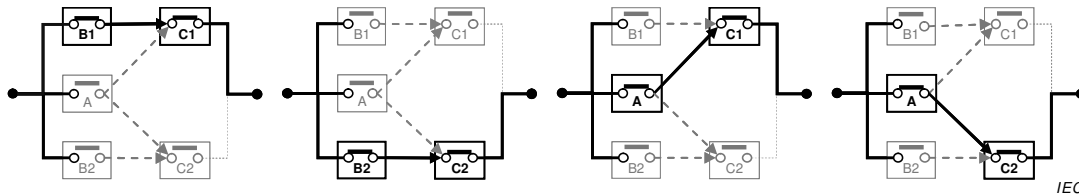


Figure B.1 – Examples of minimal tie sets (success paths)

In Figure B.1 any opening (i.e. any failure) of the closed switches will cut the circuit and lead to the system down state. All those closed switches (i.e. blocks in up states) are necessary and sufficient to have the system in up state. Those combinations are minimal and are named minimal tie sets.

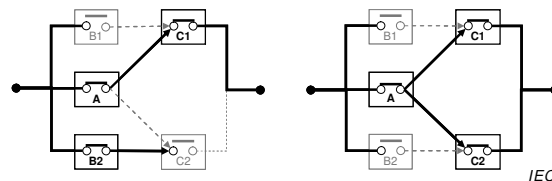


Figure B.2 – Examples of non-minimal tie sets (non minimal success paths)

In Figure B.2 some opening of the closed switches (e.g. B2 on the left or C1 on the right) will not change the system up state. All the closed switches (i.e. blocks in up states) are not necessary to have the system in up state. Those combinations are not minimal and are named non-minimal tie sets (or ordinary tie sets).

From the same Figure 15 it is also easy to identify various cut sets of this RBD and Figure B.3 and Figure B.4 show various examples of combinations of open switches corresponding to the system down state.