Improving the Efficiency of Dynamic Fault Tree Analysis by Considering Gate FDEP as Static

Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage

▶ **To cite this version:**

Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage. Improving the Efficiency of Dynamic Fault Tree Analysis by Considering Gate FDEP as Static. European Safety and Reliability Conference (ESREL 2010), Sep 2010, Rhodes, Greece. pp. 845-851. hal-00516896

HAL Id: hal-00516896
https://hal.science/hal-00516896

Submitted on 13 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improving the Efficiency of Dynamic Fault Tree Analysis by Considering Gates FDEP as Static

Guillaume Merle, Jean-Marc Roussel, and Jean-Jacques Lesage
LURPA, ENS de Cachan, Cachan, France

This paper focuses on one of the dynamic gates which are used in Dynamic Fault Trees (DFT), which is the Functional Dependency (FDEP) gate. Gate FDEP has been considered as equivalent to a set of OR gates in the literature, but this equivalence has seldom been exploited for the analysis of DFTs. In this paper, we show that in most cases, starting from a DFT including FDEP gates, the use of this static equivalence provides significant advantages for DFT Analysis.

1 INTRODUCTION

Fault Tree Analysis (*FTA*) is one of the oldest, most diffused techniques in industrial applications, for the dependability analysis of large safety-critical systems (Henley and Kumamoto 1981; Leveson 1995; Stamatelatos and Vesely 2002). *FTA* is usually carried out at two levels: a qualitative level in which the list of all the possible combinations of events that lead to the Top Event (*TE*) is determined (the *minimal cut sets*); and a quantitative level in which the probability of the occurrence of the *TE*, and of the other nodes of the tree, is calculated. One of the main restrictive assumptions in *FTA* is that basic events must be assumed to be statistically independent, and their interaction is described by means of boolean OR/AND gates, so that only the combination of events is relevant, and not their sequence. We refer to this model as *Static Fault Tree (SFT)*. Dugan et al. (Dugan, Bavuso, and Boyd 1992; Dugan, Sullivan, and Coppit 2000) proposed a new model allowing to include various kinds of temporal and statistical dependencies in the SFT model, which is the *Dynamic Fault Tree (DFT)*. The DFT is based on the definition of new gates: Priority-AND (PAND), Functional Dependency (FDEP), Warm Spare (WSP), and Sequence Enforcing (SEQ). Gate FDEP was introduced in (Dugan, Bavuso, and Boyd 1992) and allows to model common cause failures by using the concept of preemption: a trigger event allows to force dependent basic events to fail (independently of their own failure). Even though gate FDEP is most often considered as a dynamic gate (Boudali, Crouzen, and Stoelinga 2007; Stamatelatos and Vesely 2002), some authors have already mentioned that it is equivalent to a set of OR gates (Ejlali and Miremadi 2004). However, to the best of our knowledge, the influence of this equivalence on DFT analysis has never been studied. In the present paper, thanks to a DFT example, we show the influence of this equivalence on DFT Analysis. This study has been developed under the following assumptions:

- we focus on modular approaches, for which the static and dynamic parts of a DFT are analyzed by means of specific dedicated approaches;
- the results presented hold for approaches based on Continuous Time Markov Chains as well as Stochastic Petri Nets;
- this study has been validated by the tool Galileo (Dugan, Sullivan, and Coppit 2000).

The definition of gate FDEP is recalled in Section 2. The advantages of the equivalence between gate FDEP and a set of OR gates for DFT Analysis are studied in Sections 3 to 5.

2 DEFINITION OF GATE FDEP

According to (Boudali, Crouzen, and Stoelinga 2007; Dugan, Bavuso, and Boyd 1992; Stamatelatos and Vesely 2002), the FDEP gate – Functional Dependency gate – is a dynamic gate comprised of a trigger input event – either a basic event or the output of another gate of the tree – and a set of dependent basic events. Fig. 1 provides a pictorial depiction of an FDEP gate with 2 dependent basic events A and B, T representing the trigger event. When the trigger event occurs, the dependent basic events are forced to

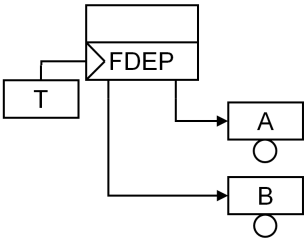

Figure 1: An FDEP gate with 2 dependent basic events A and B

occur. Numerous examples of the use of gate FDEP can be found in the literature which show that, in particular, the semantics of this gate is well adapted to model common cause failures. Indeed, the FDEP gate in Fig. 1 can allow to model a system with a component T whose failure is a common cause failure leading to the simultaneous failures of two basic events A and B, keeping in mind that basic events A and B can also fail by themselves. Gate FDEP thus offers the engineers an easy way to translate such preemptive behaviors in DFTs.

Some authors have suggested that gate FDEP could be considered as a set of OR gates. For example, the authors of (Stamatelatos and Vesely 2002), who proposed gate FDEP as a dynamic gate, also provide an alternative static representation which only includes OR gates – one for each basic event. Besides, in (Ejlali and Miremadi 2004), the time to failure of the dependent components of the FDEP gate is defined as equal to the minimum of the time left before the occurrence of the trigger event and the time left before the occurrence of the dependent event. This corresponds to the temporal definition of operator OR, based on the dates of occurrence of its input events, as described in (Merle, Roussel, Lesage, and Bobbio 2009; Merle, Roussel, Lesage, and Bobbio 2010).

3 MODULAR ANALYSIS OF DYNAMIC FAULT TREES

We are going to illustrate the usefulness of the equivalence between gate FDEP and a set of OR gates for the qualitative and quantitative analysis of DFTs thanks to the example of a computer system from (Stamatelatos and Vesely 2002) shown in Fig. 3. This DFT models the potential failure of the system in Fig. 2.

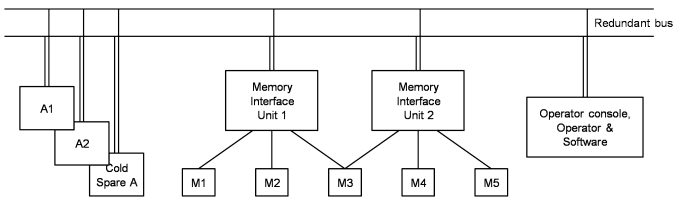

Figure 2: System corresponding to the DFT in Fig. 3

The system in Fig. 2 is composed of:

- a redundant bus subsystem including two identical buses, of which one is required for system operation;
- a redundant processing subsystem including two redundant processors $A1$ and $A2$ and a spare processor A, which can replace either upon failure. The subsystem can continue to operate until all three processors have failed;
- a redundant memory subsystem including five memory units – $M1$ to $M5$ – of which three are required, these memory units being connected to the redundant bus via two memory interface units, memory unit 3 – $M3$ – being connected to both interfaces for redundancy;
- an application subsystem including a human operator, an application (software (SW)), and a graphical user interface (hardware (HW)).

A complete description of the architecture, as well as the explanation of each considered failure, can be found in (Stamatelatos and Vesely 2002).

The DFT shown in Fig. 3 can be decomposed into 4 subtrees (Dugan, Sullivan, and Coppit 2000) according to the gates that it contains. Subtrees 3 and 4 contain static gates, only, so they can be considered as static. However, subtree 1 contains Spare gates and is hence dynamic. Finally, subtree 2 contains gates FDEP, 3-out-of-5, and AND, so it can be considered as static or dynamic, depending on the model considered for gate FDEP.

Many researchers have explored the use of divide-and-conquer approaches for analyzing such DFTs (Bobbio and Raiteri 2004; Chatterjee 1975; Dugan, Sullivan, and Coppit 2000; Dutuit and Rauzy 1996; Rosenthal 1980). For instance, in (Dugan, Sullivan, and Coppit 2000), a subtree is marked as dynamic, and solved by using Markov Chains, if it contains at least one dynamic gate. If a subtree contains no dynamic gates, it is classified as static and solved using BDD-based methods. The tool Galileo (Dugan, Sullivan, and Coppit 2000) is based on this method. The authors of (Bobbio and Raiteri 2004) also exploit the concept of modularity, a subtree that is statistically independent from the rest of the FT being denoted as a module: static modules can be analyzed by means of suitable combinatorial techniques whereas dynamic modules require a state-space analysis which is obtained by translating the dynamic module into a Generalized Stochastic Petri Net.

In both cases (Markov Chains vs. Stochastic Petri Nets), it can be useful to be able to reduce the dynamic

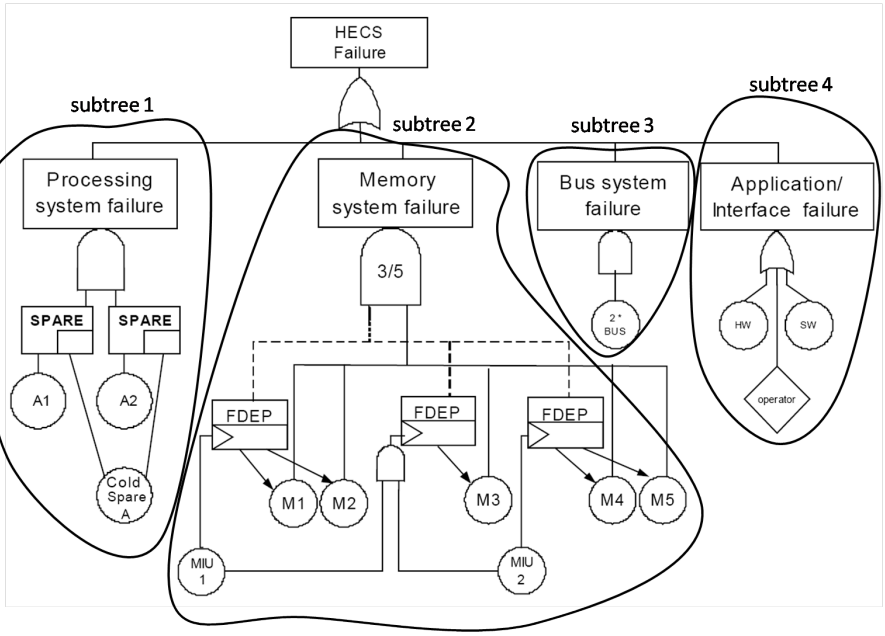

Figure 3: The 4 subtrees of the DFT in (Stamatelatos and Vesely 2002)

part of the DFT because the computational complexity of its solving is considerably higher than the computational complexity of the solving of the static part of the DFT.

The impact of the use of the equivalence between gate FDEP and a set of OR gates on the qualitative and quantitative analysis of this DFT example is analyzed in Sections 4 and 5, respectively.

4 ADVANTAGES FOR QUALITATIVE ANALYSIS

When an independent subtree of a DFT is static, BDD-based methods allow to determine the minimal cut sets of the DFT (Coudert and Madre 1993; Dutuit and Rauzy 1997). However, when such an independent subtree of a DFT is dynamic, the concept of minimal cut set must be extended to the concept of minimal cut sequence (Tang and Dugan 2004), and BDDs can no longer be used to perform the qualitative analysis of such FTs. The authors of (Tang and Dugan 2004) address this problem by using specific BDDs, denoted as Zero-suppressed BDDs, to perform the qualitative analysis of both the static and dynamic independent subtrees of DFTs. Other authors, such as (Bobbio and Raiteri 2004), convert the dynamic subtree into a Generalized Stochastic Petri Net whose occurrence graph allows to determine the minimal cut sequences of the DFT.

The fact to consider gate FDEP as dynamic or static does not have much impact on the methods used to perform the qualitative analysis of DFTs, but on the solution obtained. Indeed, in some cases, a set of minimal cut sequences can be equivalent to a single minimal cut set: for instance, if A and B are two non-repairable basic events, the set of two minimal cut sequences $\{[A,B],[B,A]\}$[1] is equivalent to the minimal cut set $A \cdot B$. As a consequence, if gate FDEP is considered as dynamic, it will provide minimal cut sequences which will be equivalent to the less numerous minimal cut sets that it will provide if it is considered as static.

Let us illustrate this possible simplification on the DFT in Fig. 3. If we consider gate FDEP as equivalent to a set of OR gates, subtree 2 contains 1 3-out-of-5 gate, 1 AND gate and 3 FDEP gates, and can hence be considered as static and equivalent to the SFT in Fig. 4.

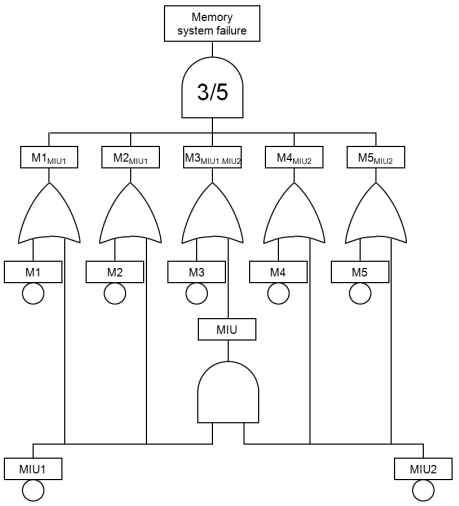

Figure 4: SFT equivalent to the subtree 2 of the DFT in Fig. 3

[1]The notation $[A,B]$ denotes the sequence of failures in which B fails after A has failed.

The subtree 2 in Fig. 3 has 74 minimal cut sequences whereas it has only 17 minimal cut sets in Fig. 4. Both results are equivalent, since a single cut set may represent a more or less big set of cut sequences. For instance, the 6 order-3 minimal cut sequences $[M1, M2, M3]$, $[M1, M3, M2]$, $[M2, M1, M3]$, $[M2, M3, M1]$, $[M3, M1, M2]$, and $[M3, M2, M1]$ are equivalent to the minimal cut set $M1 \cdot M2 \cdot M3$.

Even though both results are equivalent, minimal cut sets represent a more concise – and hence more useful – result to the practitioner than the corresponding set of minimal cut sequences. It can hence be interesting for the qualitative analysis of DFTs to consider gate FDEP as equivalent to a set of OR gates.

This equivalence also provides some advantages for the quantitative analysis of DFTs, as explained in Section 5.

5 ADVANTAGES FOR QUANTITATIVE ANALYSIS

The quantitative analysis of DFTs underlies the numerical computation of stochastic models, and consequently the use of specific software tools. For a given software, evaluation criteria are needed to compare the computational complexity for the determination of the failure probability of the *TE* for both models of gate FDEP. These criteria are studied in the following sections.

5.1 Experimental protocol

We chose to use the well-known DFT analysis tool Galileo (Dugan, Sullivan, and Coppit 2000), which exploits the concept of modularity (BDDs for static subtrees and Markov Chains for dynamic subtrees).

Since Galileo, as most DFT analysis softwares, is dedicated to personal computers, its performances are hence limited by the processor performance and the size of the RAM of the computer, since the algorithms used have an exponential complexity and the constructed state spaces of the models used are stored in the RAM, thus leading to a risk of saturation. We need evaluation criteria to evaluate the performances of the tool Galileo: the evaluation criteria that we chose for this study hence are the *computation time* and the *memory usage*. It can be noted that the computation time and memory usage needed to perform the quantitative analysis of subtrees 1, 3, and 4 of the DFT in Fig. 3 will not be impacted by the equivalence between gate FDEP and a set of OR gates since subtree 2 is statistically independent from the other subtrees. As a consequence, the impact of this equivalence on the quantitative analysis of the DFT in Fig. 3 will be limited to the quantitative analysis of subtree 2, since it is the only subtree containing FDEP gates.

Besides, the performances evaluated can be impacted by some parameters of the study. As we want to study the impact of the equivalence between gate FDEP and a set of OR gates on the quantitative analysis of DFTs, the first influent parameter coming to mind is the *number of FDEP gates*, whose impact will be studied in Section 5.3. Besides, it can be interesting to study what the impact of *time granularity* – the number of failure probabilities calculated from time zero to the mission time – is on quantitative analysis in both cases. The influence of this parameter is studied in Section 5.2.

5.2 Impact of time granularity

The computation time and memory usage obtained with Galileo to perform the quantitative analysis of the subtree 2 of the DFT in Fig. 3 when gate FDEP is considered as dynamic or static are presented in Table 1. The quantitative analysis was performed on a Pentium 4 processor with 4 GB RAM for a mission time of $T = 10,000$ hours, and with a time granularity varying from 10 to $10,000$. The memory usage of Galileo – approximately 7 MB – was removed from the memory usage obtained to get the memory usage needed for the quantitative analysis in itself.

It can be seen in Table 1 that the fact to consider gate FDEP as equivalent to a set of OR gates allows to divide the computation time for the failure probability of the *TE* by a factor of between 6 and 15. Even though the memory usage is a linear function of the time granularity in both cases, it can be divided by a factor of approximately 2 by exploiting this equivalence, thus allowing to reach a higher time granularity on the same computer.

Table 1: Computation Time CT (s) and Memory Usage MU (MB) obtained with Galileo when gate FDEP is considered as dynamic and equivalent to a set of OR gates

Time granularity	Galileo			
	dynamic		static	
	CT	MU	CT	MU
10	1	3	0	2
50	4	11	0	6
100	9	21	0	10
500	46	99	3	47
1,000	93	197	7	94
5,000	577	980	73	465
10,000	1,293	1,959	220	930

5.3 Impact of the number of FDEP gates

Let us consider the DFT in Fig. 3. Subtree 2 models the behavior of the subsystem in Fig. 5.

The analysis of this subsystem allows to understand the corresponding DFT: the failure of the Memory In-

Table 2: Variation of the Computation Time CT (s) and Memory Usage MU (MB) obtained with Galileo with the number of FDEP gates when gate FDEP is considered as dynamic or static. Two mission times are considered: $T = 100$ hours and $T = 10,000$ hours.

Order n	Number of FDEP gates	dynamic				static			
		$\Pr\{TE\}(100)$		$\Pr\{TE\}(10,000)$		$\Pr\{TE\}(100)$		$\Pr\{TE\}(10,000)$	
		CT	MU	CT	MU	CT	MU	CT	MU
2	3	0	0.5	0	0.5	0	0.4	0	0.4
3	5	0	1	0	1	0	0.5	0	0.5
4	7	0	3	2	3	0	1.5	0	1.5
5	9	2	8	5	8	0	3	0	3
6	11	4	13	13	13	0	7	0	7
7	13	11	24	41	24	0	15	1	15
8	15	28	41	104	41	2	30	4	30
9	17	51	66	193	66	11	59	17	59
10	19	82	100	300	100	25	105	35	105
11	21	125	148	445	148	35	186	56	186

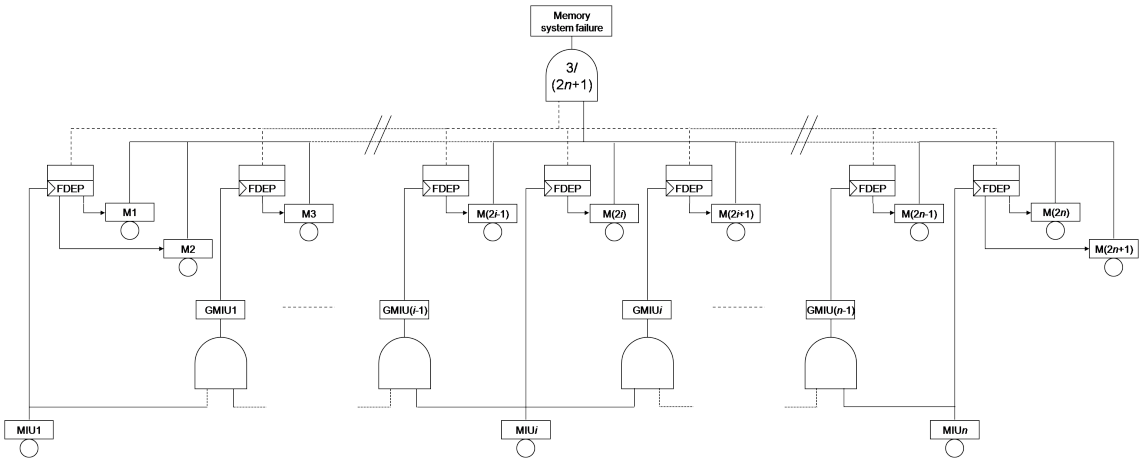

Figure 6: DFT of the order-n subsystem

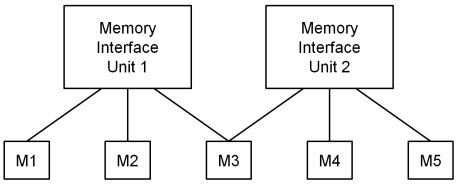

Figure 5: Subsystem modeled by subtree 2 in the DFT in Fig. 3

terface Unit 1 – $MIU1$ – will cause memories $M1$ and $M2$ to fail; in the same way, the failure of the Memory Interface Unit 2 – $MIU2$ – will cause memories $M4$ and $M5$ to fail; however, the failure of both Memory Interface Units – $MIU1$ and $MIU2$ – will be needed to cause memory $M3$ to fail.

This subsystem can be generalized to see the impact of the number of FDEP gates on the complexity of quantitative analysis. Let us consider the subsystem in Fig. 5 as the order-2 subsystem since it has 2 Memory Interface Units; its DFT contains 3 FDEP gates. The order-n subsystem with n Memory Interface Units is shown in Fig. 8. In the same way that subtree 2 in the DFT in Fig. 3 describes the potential failure of the subsystem in Fig. 5, the subsystem in Fig. 8 can be described by a DFT which is shown in Fig. 6 and contains $(2n-1)$ FDEP gates. The equivalent SFT is shown in Fig. 7 and contains $(2n+1)$ OR gates.

The computation time and memory usage needed by Galileo to perform the quantitative analysis of the DFT in Fig. 6 and of the SFT in Fig. 7 are presented in Table 2.

The results obtained in Table 2 show that the computation time varies with the mission time while the memory usage does not. However, once again, the fact to considerate gate FDEP as equivalent to a set of OR gates allows to divide the computation time for the failure probability of the *TE* by a factor of between 3 and 25, even though this equivalence becomes fruitful for a high number of FDEP gates, only.

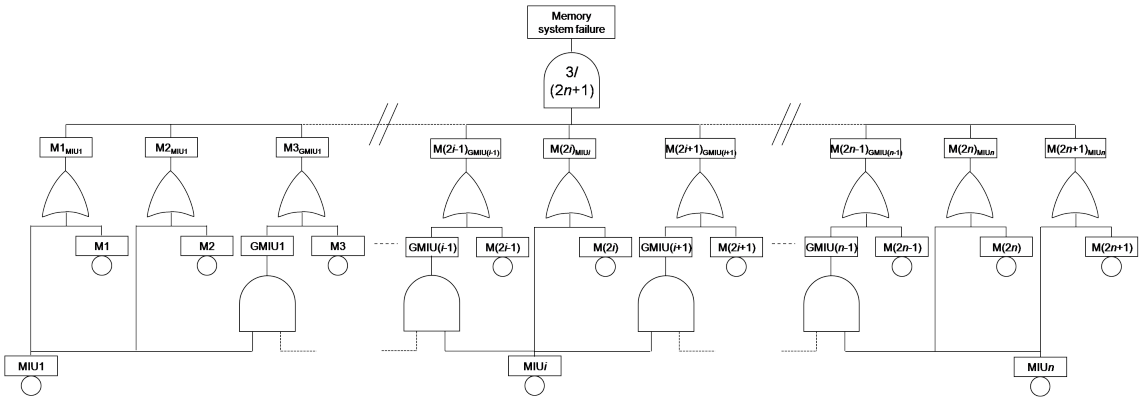

Figure 7: SFT of the order-n subsystem

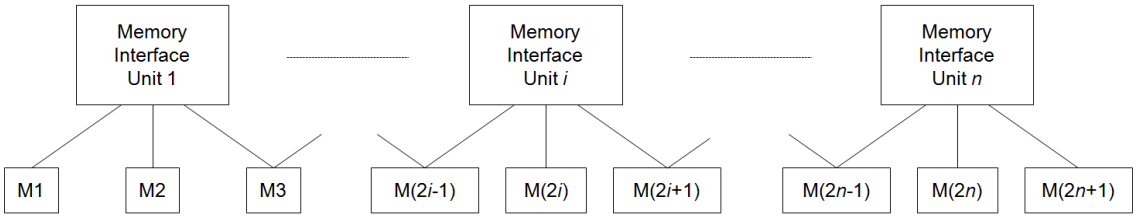

Figure 8: Order-n subsystem

6 CONCLUSION

In this paper, we have studied the influence of the equivalence between gate FDEP and a set of OR gates on Dynamic Fault Tree analysis. This influence has been illustrated on a DFT example from the literature.

The results obtained show that this equivalence has a significant influence on the computational complexity for the determination of the failure probability of the *TE* of a DFT when time granularity increases. However, this influence is somehow limited when the number of FDEP gates increases, and even least than expected, since it becomes fruitful for a high number of FDEP gates, only.

Even if this aspect is less quantifiable, we reckon that this equivalence is more fruitful on the qualitative analysis of DFTs. Indeed, as shown for the subtree 2 of the DFT example in Fig. 3, a large set of minimal cut sequences can happen to be quite difficult to exploit whereas an equivalence with a small set of minimal cuts may be more useful to the practitioner.

In any case, we have shown in this study that the fact to consider gate FDEP as a static gate can allow to convert dynamic parts of a DFT into static parts, thus simplifying both the qualitative and quantitative analysis of the DFT. It can be noted that it is in the case of dynamic subtrees containing gates FDEP only, and no other dynamic gate, that this equivalence has the most significant impact, since it allows to convert whole dynamic subtrees into static – and hence more readily analyzable – subtrees.

REFERENCES

Bobbio, A. and D. C. Raiteri (2004). Parametric Fault Trees with Dynamic Gates and Repair Boxes. In *Proceedings of the IEEE Annual Reliability and Maintainability Symposium (RAMS 2004)*, Los Angeles, CA, USA, pp. 459–465.

Boudali, H., P. Crouzen, and M. Stoelinga (2007). Dynamic Fault Tree analysis through input/output interactive Markov chains. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2007)*, pp. 25–38.

Chatterjee, P. (1975). Modularization of fault trees: A method to reduce cost of analysis. In SIAM (Ed.), *Reliability and Fault Tree Analysis*, pp. 101–137.

Coudert, O. and J.-C. Madre (1993). Fault Tree Analysis: 10^{20} prime implicants and beyond. In *Proceedings of the IEEE Annual Reliability and Maintainability Symposium (RAMS 1993)*, Atlanta, GA, USA, pp. 240–245.

Dugan, J., K. Sullivan, and D. Coppit (2000). Developing a low-cost high-quality software tool for Dynamic fault-tree analysis. *IEEE Transactions on Reliability 49*(1), 49–59.

Dugan, J. B., S. Bavuso, and M. Boyd (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability 41*(3), 363–377.

Dutuit, Y. and A. Rauzy (1996). A linear-time algorithm to find modules in fault trees. *IEEE Transactions on Reliability 45*(3), 422–425.

Dutuit, Y. and A. Rauzy (1997). Exact and truncated computations of prime implicants of coherent and noncoherent fault trees with Aralia. *Reliability Engineering and System Safety 58*(2), 127–144.

Ejlali, A. and S. Miremadi (2004). FPGA-based Monte Carlo simulation for fault tree analysis. *Microelectronics Reliability 44*(6), 1017–1028.

Henley, E. and H. Kumamoto (1981). *Reliability Engineering and Risk Assessment*. Englewood Cliffs: Prentice Hall.

Leveson, N. (1995). *Safeware: System Safety and Computers*. Addison-Wesley.

Merle, G., J.-M. Roussel, J.-J. Lesage, and A. Bobbio (2009). Algebraic Expression of the Structure Function of a subclass of Dynamic Fault Trees. In *Proceedings of the 2nd IFAC Workshop on Dependable Control of Discrete Systems (DCDS'09)*, Bari, Italy, pp. 129–134.

Merle, G., J.-M. Roussel, J.-J. Lesage, and A. Bobbio (2010). Probabilistic Algebraic Analysis of Fault Trees With Priority Dynamic Gates and Repeated Events. *IEEE Transactions on Reliability 59*(1), 250–261.

Rosenthal, A. (1980). Decomposition methods for fault tree analysis. *IEEE Transactions on Reliability R-29*(2), 136–138.

Stamatelatos, M. and W. Vesely (2002). Fault Tree Handbook with Aerospace Applications. Volume 1.1, pp. 1–205. NASA Office of Safety and Mission Assurance.

Tang, Z. and J. Dugan (2004). Minimal Cut Set/Sequence Generation for Dynamic Fault Trees. In *Proceedings of the IEEE Annual Reliability and Maintainability Symposium (RAMS 2004)*, Los Angeles, CA, USA, pp. 207–213.