Journal of Energy
VOLUME 71 Number 4 | 2022
journal homepage: http://journalofenergy.com/

On Minimal Cut Sets Representation with Binary Decision Diagrams

Reni Banov, Zdenko Šimić

***Summary* — Since their introduction in form of a canonical representation of logical functions, the Binary Decision Diagrams (BDDs) gained a wide acceptance in numerous industrial applications. This paper summarizes the properties of BDD representation of Minimal Cut Sets (MCS) of Fault Tree (FT) models most typically encountered in nuclear energetics. Cut sets from MCS are defined as paths from the top BDD node to terminal nodes in the BDD, on which a quantitative and qualitative FT analysis (FTA) is performed. The core of the FTA on the BDDs is performed with help of two fundamental algorithms, one for conditional probability evaluation and another for the selection of cut sets. The accuracy of conditional probability evaluation represents an essential feature for an unbiased quantitative analysis, such as the top event probability or the determination of event importance measures. The cut set selection algorithm is shown in a generic version introducing logical predicates for its selection criteria. As it is known, the efficiency of depicted algorithms depends only on the number of BDD nodes used for the FT representation. In order to appraise the compactness of the BDD representation of FT models, their characteristics have herein been evaluated on several real-life models from the Nuclear Power Plant Krško. The extraordinariness of the compactness of the BDD representation reflects in its ability to implement advanced dynamic analysis (i.e. what-if) of FT models. The efficiency of such an approach is recognized by commercial vendors upgrading their FT Tools to new versions by implementing BDD based algorithms.**

***Keywords* — Probabilistic Safety Assessment (PSA), Fault Tree Analysis (FTA), Binary Decision Diagrams (BDD), Minimal Cut Sets (MCS)**

I. Introduction

Introduced in the early 60's as a tool for analysing failure conditions of military systems [1], the Fault Tree Analysis (FTA) has become one of the most popular methods to deductively analyse undesired behaviour of complex engineering systems from various industries. The static Fault Tree (FT) is a directed acyclic graph (DAG) with a single top node representing a failure event under analysis. Terminal nodes at the bottom of the FT are basic events representing a component failure occurrence and are considered relevant for the analysis. The intermediate nodes are conditions under which the basic events propagate their occurrence to the top node. An example of a simple FT is given in Figure 1. In a general view to FTA, we differentiate two types of static analysis: the qualitative and the quantitative [2]. Under the qualitative analysis the FT is typically evaluated to find minimal cut sets, minimal path sets, and common cause failures. Quantitative analyses are performed numerically with the goal of computing various reliability measures, like system availability, mean time between failures, component importance measures, and others.

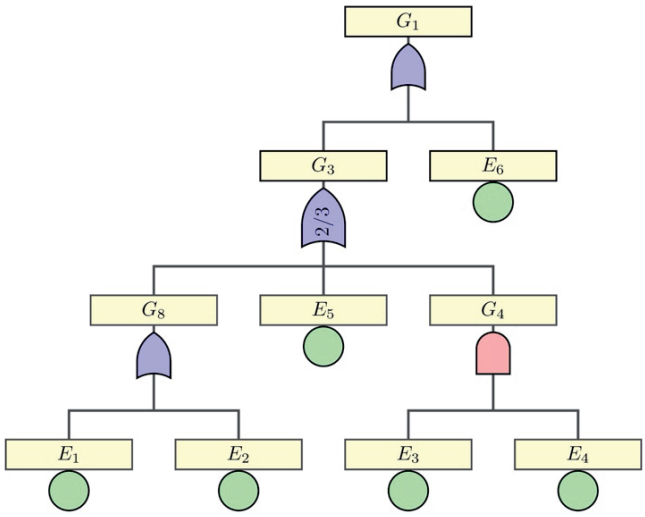

Fig. 1. Fault Tree example

It should be brought to attention that the FT structure represents a large Boolean function which depends on the occurrence of basic events, thereby allowing their minimization to find representation in form of minimal cut sets (MCSs). The conventional approach to FTA relies on a process of determination of minimal cuts sets from the FT structure by applying a simplification rule according to the Boolean laws. The two most common conventional approaches are based on top-down or bottom-up rewritings of logical formulas defined by intermediate FT nodes. Both approaches are computationally intensive and are resource demanding, and may, thereby, be applied only to determine the most significant minimal cuts based on probability values or their size. Lately, the new techniques are constructed on Binary Decision Diagrams (BDDs) [3, 4] and Monte Carlo [5] simulation methods. The BDD method of minimization of Boolean functions seems to be more powerful than Monte Carlo methods, though it depends on the knowledge of a good basic event order to achieve supremacy [6]. The advantage of

(Corresponding author: Reni Banov)
Reni Banov is with the University of Applied Sciences (TVZ) Zagreb, Croatia (e-mail: reni.banov@tvz.hr)
Zdenko Šimić is with the Energy Institute Hrvoje Požar (EIHP) Zagreb, Croatia (e-mail: zsimic@eihp.hr)

Monte Carlo simulation methods is that they can be easily applied to the dynamic fault tree (DFT) analysis.

However, finding the minimal form of any Boolean function is a NP-complete problem, even for the simplest case of monotonic Boolean functions. The time and space complexity of a problem requires a novel data structure to represent Boolean functions. Normally, Boolean functions are represented by truth tables or logical expressions, but the BDD structure, i.e., a variant of DAG with two terminal vertices, are far more efficient for implementation. The BDDs for Boolean functions are derived from the Shannon identity

$$f(\boldsymbol{x}) = \left(x_i \wedge f(\boldsymbol{x}; 1_i)\right) \vee \left(\neg x_i \wedge f(\boldsymbol{x}; 0_i)\right) \quad (1)$$

applied recursively to each function in expansion. Each Shannon step generates a part of a full binary tree with vertices structured, as shown in Figure 2, down to the bottom of the tree with two terminal vertices representing Boolean values {**0**,**1**}.

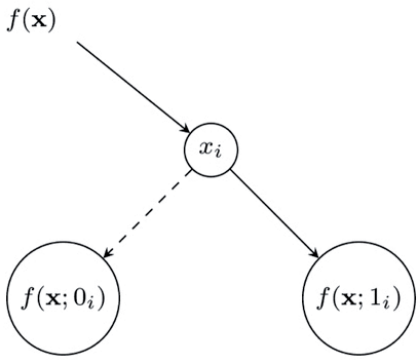

Fig. 2. Shannon identity step as binary tree

The Shannon identity with BDDs is commonly expressed by means of the If-Then-Else (*ite*) construction, for instance, the tree structure of the function from Figure 2 is written as

$$ite(x_i, f(\boldsymbol{x}; 1_i), f(\boldsymbol{x}; 0_i)) \quad (2)$$

If an order of variables is the same (preserved) on each path from any vertices down to terminal vertices, the BDD is denoted as *ordered* BDD [7]. During the Shannon expansion the vertices for each unique Boolean function are generated only once, thereby making an ordered BDD reduced and ensuring the uniqueness of representation, i.e., the canonical representation of the Boolean function. The BDD structure allows an efficient implementation of usual logical operations, which makes it a suitable tool for manipulating Boolean functions.

II. BDD Method for MCS Set

From the very beginning of the application of fault tree analysis in nuclear energetics it has clearly come to mind, that a more accurate insight into the reliability of the observed system relies on the understanding of the complete or, at least, the most significant parts of failure sets (minimal cut sets MCSs). However, the determination of MCSs turns out complex even with very simple systems modelled by a coherent fault tree, dealing with at least two hard problems. The first, being the time complexity of algorithms employed for the determination of the complete or partial set of MCSs, while the second relates to a space complexity of the same sets recording. More recently binary decision diagrams (BDDs) have been developed, enabling an indirect recording of fault trees by applying indicator variables for the component failure state within the system.

The basic idea behind the BDD method is to define an indicator variable which acquires the logical value zero (false) if the basic event does not occur, and inversely, the logical value one (true) if the basic event occurred. The probability of the basic event occurrence is thereby associated with the probability of true occurrence of the indicator variable. In this way the Bernoulli random variable is assigned to the basic event. Once the logical function represented by the FT is converted to a BDD representation, the BDD based method [3] can be used to find its minimal disjunctive form which represents a MCS set of the coherent FT. Figure 3 shows the BDD representation of the complete MCS set of the FT example from Figure 1. The dotted arrow line marks that the MCS does not include the event originating from the line, while the full arrow line stands for the event included in MCS. The full MCS set is defined by all paths from the top node ending at the node with value one (true). It is worthwhile mentioning that non-coherent FTs can be treated similarly with Zero Decision Diagrams (ZDDs) which are a variant of BDD supporting combinatorial sets [8].

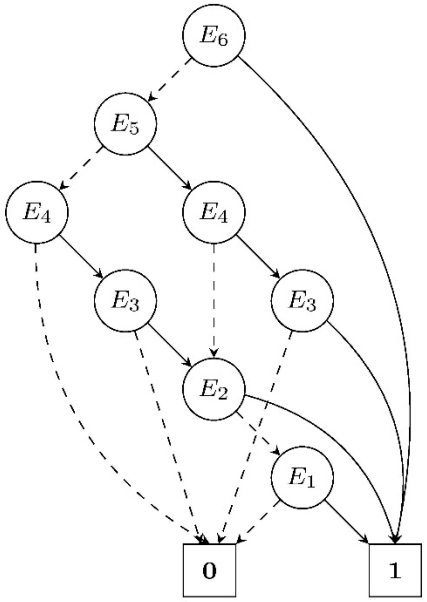

Fig. 3. MCS as BDD tree

Once the BDD of the MCS set is created, an analysis can be performed on indicator variables with algorithms specifically written for the BDD structure. The qualitative analysis relies on the selection of MCS sets according to specific criteria. This kind of analysis is easily performed based on the algorithm which can select a subset of MCSs from BDD with predicates applied to the indicator variables. The underneath algorithm (Figure 4) implements a subset selection from the full MCS set represented by the BDD structure defined on indirect variables.

The essence of the algorithm is to select from a full MCS set only cut sets for which the predicate results in true value on indirect variables. With a new minimal cut set the decision is rather simple, namely, as soon as the terminal node with value one is reached, the predicate on truth values of indirect variables traversed through the path can be applied. This part of the algorithm is entailed in lines 9-14. Not having reached the terminal node means that we still dwell on a node determined by a single indirect variable (a single basic event). Subsequently, we can pursue the traversal of

the BDD graph via the branch containing that variable (full arrow line), assuming that the predicate responded in value one (true) for the part of the cut set found. Contrary to that, on a zero (false) responded value the traversal is carried on with the branch not containing that variable (dotted arrow line). This part of the algorithm from Figure 4 is displayed in lines 16-20. The lines 6-8 from the algorithm represent paths leading to the zero terminal node, i.e., to the node indicating that the minimal cut set has not been found through the path. It is important to mention that the predicate for the cut set selection must return a logical value even if only a part of the minimal cut set stands. For example, predicates conforming to that criteria are typical predicates used for the analysis, such as a maximal number of basic events in a cut set or a minimal cut set probability value.

The cut set probability evaluation is performed with the assumption that basic events are independent, in other words, the probability value of a cut set equals to the product of probabilities of basic events contained in the cut set. Basically, by introducing line 17 in the algorithm a significant reduction of the traversal has been performed resulting in a reduction of the execution time for the minimal cut set selection. What is more, by introducing the concept of predicates for the cut set selection we have achieved a flexibility for the creation of complex selection criteria since logical expressions defined by predicates can be combined with common logical operators. For example, it is rather easy to specify criteria containing cut sets which bear a specific number of basic events and meet either conditions of a minimal probability value or conditions containing a specific basic event. In this way we have achieved a significant flexibility to perform a highly specific qualitative and quantitative FT analysis.

Input: $MCS(v_m, mcs, F, cs)$
Output: $mcs \leftarrow MCS(v_m, F)$

```
1  v_m ∈ V_mcs                                          // MCS set BDD top node
2  mcs = {cs_i}                                         // MCS subset selected with predicate F
3  F: {E_i} → 𝔹                                         // Selection criteria as predicate
4  cs = {E_i}                                           // Current cut set
5  begin
6      if value(v_m) = 0 then
           /* Terminal node 0                                                    */
8          return
9      else if value(v_m) = 1 then
           /* Terminal node 1                                                    */
10         if F(cs) = 1 then
               /* Predicate is fullfiled with current cut set cs                 */
12             mcs = mcs ∪ cs
14         return                                       // Terminal node 1 ends current cut set
15     else
           /* E_index(v_m) basic event indirectly specified with index(v_m)      */
16         if F(cs ∪ {E_index(v_m)}) = 1 then
18             return MCS(high(v_m), mcs, F, cs ∪ {E_index(v_m)})
20         return MCS(low(v_m), mcs, F, cs)
```

Fig. 4. MCS subset selection

The second basic algorithm (Figure 5) represents the calculation procedure of the conditional probability from the paths in the BDD graph. Analogously to the previous algorithm the BDD structure is traversed in a depth first manner and conditional probabilities are calculated from indicator variables encountered on the traversal path. Whenever we reach terminal nodes we need to return the probability value confirmed by the truth value of terminal nodes, thus, for terminal node one the value 1.0 is returned, while for terminal node zero the value 0.0 is returned. This part of the algorithm is set forth in lines 5-10. Once an intermediate node is reached, by checking the truth value of the indicator variable from the parameter set (σ), we may decide on the continuance of the BDD traversal. Thus in lines 14-16 the traversal is continued in case of a zero (false) value indicator variable, i.e., in this case we are calculating the conditional probability provided that the basic event associated with the indicator variable did not occur. Similarly to that, in lines 17-19 the conditional probability is calculated assuming that the particular basic event has occurred. In all other cases we continue with a recursive calculation of the conditional probability by traversing the BDD structure on left and right branch nodes (line 22). It is essential to mention that the significant algorithm optimisation may be achieved by saving the intermediate result of the conditional probability calculation for that node. In this way a multiple calculation of conditional probabilities for the same nodes encountered during the traversal is avoided, which at the end, results in time efficient computation.

Input: $\Pr(v_f, \sigma)$
Output: $p \leftarrow P(\sigma(f) = 1)$
Data: $\sigma(f)$ set of indicator variables with known state

```
1  begin
2      if value(v_f) = 0 then
4          return 0                                     // Terminal node 0
5      else if value(v_f) = 1 then
7          return 1                                     // Terminal node 1
8      else
           /* probability of basic event E_index(v_f)                            */
10         p_i ← P(E_index(v_f) = 1)
11         if σ(index(v_f)) = 0 then
13             return (1 − p_i) · Pr(low(v_f), σ)       // Basic event E_index(v_f) = 0
14         else if σ(index(v_f)) = 1 then
16             return p_i · Pr(high(v_f), σ)            // Basic event E_index(v_f) = 1
17         else
               /* Unknown state of the basic event E_index(v_f)                  */
19             return
                   Pr(low(v_f), σ) + p_i · [Pr(high(v_f), σ) − Pr(low(v_f), σ)]
```

Fig.5. Indirect evaluation of conditional probability from BDD

Now, a qualitative and quantitative analysis on a fault tree model may be carried out with BDDs by applying known algorithms for the determination of a minimal disjunctive normal form of the logical function presented by the coherent fault tree, which represents the logical recording of a set of minimal cuts.

III. Results and Discussion

The previous algorithms are implemented in the C/C++ programming language and their correctness and accuracy has been tested on FT models from the nuclear power plant Krško. For the implementation of the BDD algorithm it was necessary to find a good basic event order by which the BDD representation of the FT may be traced. The following table brings characteristics of FT models (column *B.E.* stands for the number of basic events, column *Gates* is the number of intermediate events) utilized in the testing as well as properties of the BDD representation of their complete MCS set.

Table I
Results on NEK FT test models

FT	B.E.	Gates	BDD	MCS	Ratio MCS/BDD
acp	409	674	4.583	228.242.636	49.802
chrgr	438	695	68.904	14.840.731.139.897	215.382.722
dcp	447	706	77.595	152.148.878.846.392	1.960.807.769
efw	692	957	265.401	1.769.960.840.506.752	6.669.005.921
hpsi	674	940	22.902	371.554.422.700	16.223.667
lpsi	525	760	26.754	479.582.239.771	17.925.627
sw	444	720	140.486	58.952.275.075.664	419.630.960
cored1	1319	1279	1.951.673	69.273.024.997.243.046	35.494.176.020
cored2	1377	1633	16.524.072	2.436.058.751.633.933.343	147.424.844.895

Reni Banov, Zdenko Šimić, On Minimal Cut Sets Representation with Binary Decision Diagrams, Journal of Energy, vol. 71 Number 4 (2022), 12–15
https://doi.org/10.37798/2022714420

The number of cut sets in MCS set (column *MCS*) for the tested FT models ranges between 10^8 and 10^{18}, while the BDD size (column *BDD*) for the most complex model (cored2) comes to 16.5 million nodes. Since the nodes are represented with a structure of 32 bytes sized we can conclude that the full MCS set for the most complex model shall need approx. 500MB RAM memory. It is hard to conceive how much memory it would take for conventional FTA programmes to represent a complete MCS set of the cored2 model.

Apart from the compact representation, the BDD structure allows an efficient execution of the mentioned algorithms; e.g., the selection of the MCS subset according to predicate criteria for a cut set length equalling to 5 basic events lasts for approx. 2 seconds on a desktop PC with 8GB RAM and an Intel i5 processor. The execution time of the algorithm is predominantly influenced by the selection of the traversal branch which does not meet the predicated condition, i.e., on the branch with a basic event for which the predicate returns a false value in the early traversal phase (see line 16 of algorithm in Figure 4).

Also, the top event conditional probability calculation by means of the algorithm from Figure 5 takes on an average less than one second even for the probability of a top event without any condition on indicator variables. This is the most complex case, since BDD traversals are performed through each node. However, once the result for every node is saved (a single double precision number) we can reuse the calculated result which significantly speeds up the calculations, as by this the complexity of the algorithm becomes proportional to the number of nodes in the BDD. Effectively, for the most complex model we achieved the worst case complexity of order double precision operations. The *n* in the complexity order represents the number of nodes from the BDD. The above written indicates an outstanding compact BDD representation (column *Ratio MCS/BDD*) of the MCS set and a remarkably efficient implementation of the analysis algorithm. The respective column indicates the average quantity of paths going through a BDD node.

Recently [9], besides the BDD representation compactness the results of the quantitative analysis performed on BDDs were thoroughly compared to the results obtained with conventional FTA tools. The authors compared the results obtained by means of these two techniques (conventional and BDD) on the Liebstatdt NPP model (KKL) and found some interesting outcomes. For example, they established that a "*substantial reduction in CDF/FDF was achieved for KKL PSA model*" signifying that the application of the BDD approach may have potential on the reduction of risk metrics in other models, too. It is worthwhile mentioning that the BDD quantitative analysis approach results in exact values, thereby not having any biases commonly occurring with conventional approaches.

IV. Conclusion

The preparation of the basic event ordering for the application of BDD methods makes the most important task of the analysis based on the BDD structure. The ordering procedure alone is the principal time consuming task; luckily it is performed only once and does not have to be repeated for other calculations. Along with the ordering, the MCS set is also computed once and needs not to be repeated unless the structure of the FT model has been changed. The exceptional compactness of minimal cut set recordings gained by the BDDs technique ensures the recording of a complete set of MCSs. The complete MCS set is defined by a logical function on indicator variables defined from the FT model. Once the complete MCS set has been found, the analysis is repeatedly performed by changing the conditions. For example, changing the probability of a basic event occurrence or defining different selection predicates enables a repeated analysis without MCS set re-determination.

The most distinguished advantage of the BDD based FTA is its compact representation and the fact that the qualitative and quantitative analysis can be performed on complete MCS sets. Actually, the numerical precision of the calculations does not depend on the number of cut sets in the MCS set entirely unlike conventional FTA approaches that must re-compute a part of the MCS set and perform analysis thereon.

Another important feature of BDD based algorithms is that their complexity is proportional to the number of BDD nodes and by this, they do not depend on the number of cut sets in the complete MCS set. Thus, not only do BDDs show (under the condition of an appropriate variable order) an acceptable time complexity for the implementation of algorithms for determining and analysing MCSs but also enable a compact recording of complete or partial sets of MCSs singled out in that way. Along with this, the compact BDD representation allows the development of new and improved analysis techniques since a complete MCS set is available for the implementation of such algorithms. This circumstance opens new prospects for further research and development of BDD analysis methods, especially in the field of nuclear energetics which utilizes the most complex FT models.

References

[1] C.A. Ericson, "Fault Tree Analysis – a history", *In Proceedings of the 17th International System Safety Conference*, Orlando, Florida, USA, 16-21 August 1999, pp. 1 – 9

[2] E.J.J. Ruijters, M.I.A. Stoelinga, "Fault Tree Analysis: A survey of the state-of-the-art in modeling, analysis and tools", CTIT TR-CTIT-14-14, Centre for Telematics and Information Technology, University of Twente, NL., 2014.

[3] A.B. Rauzy, "New algorithms for fault tree analysis", *Reliability Engineering & System Safety*, Vol. 40, No. 3, pp. 203–211, 1993.

[4] R. Remenyte-Prescott, J.D. Andrews, "An enhanced component connection method for conversion of fault trees to binary decision diagrams", *Reliability Engineering & System Safety*, Vol. 93, No. 10, pp. 1543–1550, 2008.

[5] K. Durga Rao, V. Gopika, V.V.S. Sanyasi Rao, H.S. Kushwaha, A.K. Verma, A. Srividya, "Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment", *Reliability Engineering & System Safety*, Vol. 94, No. 4, pp. 872–883, 2009.

[6] R. Banov, Z. Šimić, D. Grgić, "A new heuristics for the event ordering in binary decision diagram applied in fault tree analysis", *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, Vol. 234, No. 2, pp. 397–406, 2020.

[7] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transaction on Computers*, Vol. 38, No. 8, pp. 677–691, 1986

[8] O. Coudert, J.C. Madre, "Metaprime: An interactive fault-tree analyzer", *IEEE Transaction on Reliability*, Vol. 43, No. 1, pp. 121–127, 1994

[9] P. Zvoncek, O.Nusbaumer, "Comparison of MCUB and MCS BDD Fault Tree Solution Algorithms using Liebstadt Nuclear Power Plant Model", *PSAM 14*, Los Angeles, CA, USA, 16-21 September 2018