Nuclear Reactor Safety Optimization: Boolean Algebra Reduction in Fault Tree Analysis using the Quine-McCluskey Algorithm

Mulky Siraj Firizqi - 13525069
Informatics Engineering Study Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Ganesha 10 Bandung Street
E-mail: mulkysiraj0412@gmail.com , 13525069@std.stei.itb.ac.id

Abstract*—Nuclear reactor safety requires rigorous risk analysis, traditionally executed through Fault Tree Analysis (FTA). However, identifying Minimal Cut Sets (MCS) in complex systems presents significant computational challenges due to the exponential growth of Boolean expressions. This paper presents a computational implementation of the Quine-McCluskey (QMC) algorithm via a Python-based script to automate Boolean logic reduction in a seven-variable Emergency Core Cooling System (ECCS) model. By evaluating a 128-row truth table, the model initially identified 77 distinct failure minterms. The automated QMC pipeline deterministically reduced this 77-term unminimized canonical Sum-of-Products (SOP) into exactly 5 MCS. This optimization achieved a 97.96% reduction in the Boolean literal count (from 539 to 11 literals) and drastically condensed the computational time complexity for future quantitative probability calculations from $O(2^{77})$ to $O(2^{5})$ operations. From an engineering perspective, the extracted MCS successfully isolated critical system vulnerabilities, analytically proving the systemic threat of a 2nd-order Station Blackout scenario and identifying the water storage sensor as a structural bottleneck. Ultimately, this study validates that the QMC algorithm is not merely a theoretical construct of discrete mathematics, but a robust and computationally tractable mechanism for industrial safety engineering.

***Keywords*—Boolean Algebra, Fault Tree Analysis (FTA), Quine-McCluskey, Minimal Cut Set, Nuclear Reactor Safety, Minimal Cover, Prime Implicant**

I. Introduction

The modern energy and nuclear industries operate reactor systems with high pressures, extreme temperatures, and hazardous materials, the failure of which can result in catastrophic nuclear accidents and radiological contamination. Tragedies such as the Three Mile Island (1979), Chernobyl (1986), and Fukushima Daiichi (2011) nuclear accidents serve as reminders that inadequate safety analysis can result in massive loss of life and material damage [6], [7]. Fault Tree Analysis (FTA) has been adopted as an industry standard by bodies such as IEC 61025 and the NASA Fault Tree Handbook for conducting qualitative and quantitative system reliability analysis [3], [4], [12]. FTA represents cause and effect relationships in the form of a logic tree using AND and OR gates, which naturally form Boolean Algebraic expressions. A fundamental challenge arises when the fault tree grows to tens or hundreds of basic events. The resulting Boolean expressions become extremely complex, so identifying the minimal cut set, the core of qualitative FTA analysis, requires systematic reduction techniques.

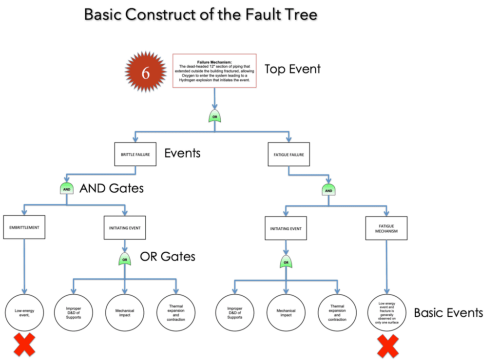

Fig. 1.1. Illustration of Fault Tree Analysis

(Source:https://bluedragonrootcause.com/fault-tree-analysis/)

This is where Discrete Mathematics, specifically Boolean algebra and combinatorial optimization algorithms, becomes an irreplaceable mathematical foundation. The Quine-McCluskey (QMC) algorithm, developed by Willard Van Orman Quine in 1952 and refined by Edward J. McCluskey in 1956, provides a deterministic and computationally implementable tabulation procedure [1], [2]. This algorithm guarantees finding the minimal form (minimal sum-of-products) of any Boolean function, unlike the Karnaugh Map method, which is practically limited to functions with six or fewer variables [2]. Therefore, this paper aims to bridge the gap between theoretical discrete mathematics and industrial safety by applying the QMC algorithm to evaluate a seven variable nuclear reactor Fault

Tree Analysis [14]. Furthermore, this study integrates a computational approach by implementing a Python-based script to automate the Boolean reduction process, demonstrating the practical scalability of the algorithm in engineering applications.

II. THEORETICAL BASIS

A. *Boolean Algebra*

Boolean Algebra, first proposed by George Boole in his work "The Mathematical Analysis of Logic" (1847), is an algebraic system that operates on two values: 0 (False) and 1 (True) [5]. This system is defined by the set B={0,1} with three fundamental operations: conjunction (AND, ·), disjunction (OR, +), and complement (NOT, ′). Boolean Algebra is widely used in the process of analyzing digital circuits such as Integrated Circuits also includes Fault Tree Analysis for the safety of reactors.

The basic operations of Boolean Algebra are:

1. (+)
   i. 0 + 0 = 0
   ii. 0 + 1 = 1
   iii. 1 + 1 = 1
2. (.)
   i. 0.0 = 0
   ii. 0.1 = 0
   iii. 1.1 = 1

The basic laws of Boolean Algebra are:

1. Law of Identity
   i. a + 0 = a
   ii. a.1 = a
2. Law of Idempotence
   i. a + a = a
   ii. a.a = a
3. Law of Complement
   i. a + a’ = 1
   ii. a.a’ = 0
4. Law of Domination
   i. a.0 = 0
   ii. a + 1 = 1
5. Law of Involution
   i. (a’)’ = a
6. Law of Absorption
   i. a + a.b = a
   ii. a. (a + b) = a
7. Commutative Law
   i. a + b = b + a
   ii. a.b = b.a
8. Distributive Law
   i. a.(b + c) = a.b + a.c
   ii. a + (b.c) = (a + b).(a + c)
9. Associative Law
   i. a + (b + c) = (a + b) + c
   ii. a.(b.c) = (a.b).c
10. De Morgan’s Law
    i. (a + b)’ = a’.b’
    ii. (a.b)’ = a’ + b’
11. Law of 0/1
    i. 0’ = 1
    ii. 1’ = 0

Boolean functions can also be represented as logic circuits. Simple logic gates created from these laws and operations are:

1. AND Gate

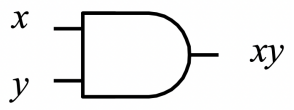

Fig. 2.1.1. AND Gate

(Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/10-Aljabar-Boolean-(2026)-bagian1.pdf)

Table 2.1.1. AND Logic Gate Results

Input		Output
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

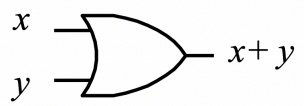

Fig. 2.1.2. OR Gate

(Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/10-Aljabar-Boolean-(2026)-bagian1.pdf)

Table 2.1.2. OR Logic Gate Results

Input		Output
x	y	x + y

0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

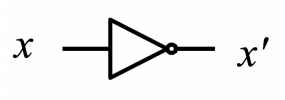

Fig. 2.1.3. NOT Gate

(Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/10-Aljabar-Boolean-(2026)-bagian1.pdf)

Table 2.1.3. NOT Logic Gate Results

Input	Output
x	x’
0	1
1	0

Other Boolean operators such as XOR, NAND, NOR, and XNOR exist and are widely used in digital circuit design. However, Fault Tree Analysis primarily models system failures using AND and OR relationships, with occasional use of complement operations. Therefore, this paper focuses only on these fundamental operators. In the context of Fault Tree Analysis (FTA), Boolean variables represent basic component events. A value of 0 indicates a normal component function, while a value of 1 indicates a failure state. The relationships between these failures are mapped using logic gates.

B. Fault Tree Analysis

Fault Tree Analysis (FTA) is a top-down, deductive failure analysis method used to identify the root causes of an undesired system-state, referred to as the "Top Event" [12]. The system failure paths are graphically represented using a tree-like structure composed of logical gates. In the context of risk assessment, FTA translates physical component dependencies into deterministic Boolean expressions. To properly model a system's reliability, standard graphical symbols specified in safety guidelines (such as NUREG-0492) are used as follows:

1) Top Event / Intermediate Event: Represented by a Rectangle. It signifies a fault state that occurs due to the combination of other antecedent faults propagating through logic gates.
2) Basic Event: Represented by a Circle. It signifies the lowest level of identifiable component failure that requires no further logical breakdown.
3) AND Gate: Represents a condition where the output fault occurs if and only if all input events co-occur simultaneously. Mathematically, this corresponds to the Boolean multiplication (·) operator.
4) OR Gate: Represents a condition where the output fault occurs if at least one of the input events occurs. Mathematically, this corresponds to the Boolean addition (+) operator.

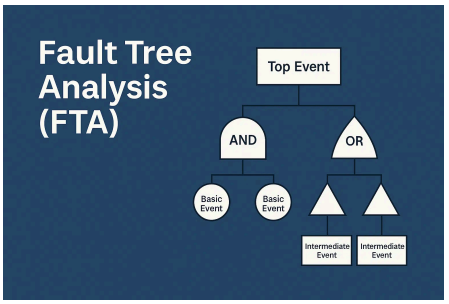

Fig. 2.2. Deductive Structure of a Fault Tree Diagram

(Source: https://www.hsestudyguide.com/fault-tree-analysis-fta/)

The core objective of qualitative FTA is identifying the Minimal Cut Set (MCS). An MCS is defined as the smallest set of basic events that, if all occur simultaneously, will guarantee the top event [13]. Formally, given a cut set K of an FTA, K is considered minimal if and only if no proper subset $K' \subset K$ is also a valid cut set. Mathematically, extracting the MCS is equivalent to finding the prime implicant cover of the Boolean expression that represents the Top Event.

To illustrate the mathematical extraction of an MCS, consider a theoretical sub-system where a Top Event (T) occurs if a fault tree dictates the logic: (A OR B) AND (A OR C). The initial Boolean expression derived from the logic gates is:

$$T = (A + B)\cdot(A + C)$$

The process to find the MCS involves applying Boolean Algebraic laws to reduce the expression into its simplest Sum-of-Products (SOP) form:

1) Step 1 - Expression Expansion (Law of Distribution): Multiply the terms to find all possible combinatorial cut sets

$$T = (A.A) + (A.C) + (B.A) + (B.C)$$

2) Step 2 - Elimination of Redundancies (Law of Idempotence): Since a variable ANDed with itself is just the variable (A.A=A), the expression simplifies to:

$$T = A + AC + AB + BC$$

Note: At this stage, the identified cut sets are {A},{A,C},{A,B}, and {B,C}. However, they are not yet minimal because some sets are proper subsets of others.

3) Step 3 - Reduction to Minimal Form (Law of Absorption): Applying the absorption rule (A + AB = A), we understand that if basic event A alone is sufficient to trigger the Top Event, then the combinations AC and AB are redundant.

$$T = A(1 + C + B) + BC$$

$$T = A(1) + BC$$

$$T = A + BC$$

The final Boolean expression yields exactly two Minimal Cut Sets: {A} and {B,C}. This result mathematically proves that system failure can be mitigated by prioritizing the prevention of the single point of failure (A), or by preventing the simultaneous failure of components B and C.

C. Boolean Canonical Forms (Truth Table, Minterms, SOP, and POS)

Before applying minimization algorithms, logical expressions derived from Fault Tree Analysis must be standardized into canonical forms using truth tables. A truth table mathematically maps all possible binary states (0 or 1) of the input variables to the system's output. To illustrate, consider a hypothetical 3-variable safety system (A,B,C) governed by the unsimplified logical expression

$$F = C \cdot (A' + B)$$

By evaluating this equation for all $2^3 = 8$ possible combinations of inputs, we generate the following truth table mapping:

Table 2.3.1. Truth Table of Logical Expression F

A	B	C	Output (F)	Designation	Term Type
0	0	0	0	M_0	Maxterm
0	0	1	1	m_1	Minterm
0	1	0	0	M_2	Maxterm
0	1	1	1	m_3	Minterm
1	0	0	0	M_4	Maxterm
1	0	1	0	M_5	Maxterm
1	1	0	0	M_6	Maxterm
1	1	1	1	m_7	Minterm

From this evaluated truth table, Boolean functions are expressed in two primary standard canonical forms:

1) Minterms and Sum of Products (SOP): A minterm (denoted as m) is a product (AND) term that includes all variables of a function exactly once, either in their true (1) or complemented (0) form. A minterm exclusively represents the rows where the output evaluates to 1 (True). A Boolean function can be expressed as the logical sum (OR) of its minterms, which is known as the Sum of Products (SOP) form. For the table above, the SOP focuses on the conditions that cause the system to fail (F=1):

$$F(A, B, C) = \sum m(1, 3, 7) = A'B'C + A'BC + ABC$$

2) Maxterms and Product of Sums (POS): Conversely, a maxterm (denoted as M) is a sum (OR) term containing all variables, representing the rows where the output evaluates to 0 (False). A Boolean function can also be expressed as the logical product (AND) of its maxterms, known as the Product of Sums (POS) form. For the table above, the POS focuses on the safe states (F=0):

$$F(A, B, C) = \prod M(0, 2, 4, 5, 6) = (A + B + C) \cdot (A + B' + C) \cdot (A' + B + C) \cdot (A' + B + C') \cdot (A' + B' + C)$$

In engineering safety and failure probability (such as Fault Tree Analysis), the analysis heavily prioritizes the Sum of Products (SOP) form. This is because reliability engineers need to isolate the exact combinations of component failures (minterms) that trigger a top event disaster. Consequently, algorithms designed for fault logic reduction, such as the Quine-McCluskey method, operate inherently on this SOP architecture.

D. Quine-McCluskey Algorithm

The Quine-McCluskey (QMC) algorithm is a tabulation method for simplifying Boolean functions that guarantees a minimal Sum-of-Products (SOP) form [1], [2]. While the previous FTA example demonstrated the reduction of only three variables (A,B,C), which can easily be simplified using fundamental Boolean Algebraic laws or visual methods like the Karnaugh Map (K-Map), these manual approaches become obsolete as system complexity scales.

In real-world nuclear reactor safety systems, fault trees often involve tens of interrelated component variables. When the number of variables exceeds four, manual algebraic reduction is highly error-prone, and K-Maps lose their visual intuitiveness, becoming virtually impossible to construct beyond six variables [2]. Therefore, a systematic, computationally programmable approach is fundamentally required to prevent catastrophic human errors in safety analysis. To overcome this limitation, the QMC algorithm is utilized. This algorithm operates in two main phases:

1) Phase I - Prime Implicant Formation: All minterms (rows with a value of 1) and don't-care terms are grouped based on the number of '1' bits in their binary representation. Then, iteratively merge pairs of minterms that differ by only one bit (the adjacency principle). This process continues until no more pairs can be merged. Implicants that cannot be further merged are called prime implicants.
2) Phase II - Essential Prime Implicant Cover Selection: A prime implicant chart is created showing which minterms are covered by each prime implicant. An essential prime implicant is the only one that covers a particular minterm and, therefore, must be included in the solution. After the essential prime implicant is selected, a column elimination process (Petrick's method or a greedy approach) is carried out to select the most efficient non-essential prime implicant in covering the remaining minterms [16].

To demonstrate the deterministic nature of the QMC algorithm, consider a 4-variable Boolean function with the following minterms:

$$F(W,X,Y,Z) = \sum m(0,2,6,7,8,10,14,15)$$

1) Phase I (Prime Implicant Formation): Minterms are initially grouped by the number of '1's in their binary representation. Iterative pairing is then performed (combining terms that differ by only one bit, replacing the differing bit with a dash '-') until no further pairings are possible.

Table 2.4.1. Phase I (Prime Implicant Formation)

Group (1s)	Initial Minterms	1st Merge (1-bit diff)	2nd Merge (2-bit diff) → Prime Implicants
0	m_0(0000)	(0,2)⇒00−0 (0,8)⇒−000	P_1(0,2,8,10) ⇒ −0−0 (X′Z′)
1	m_2(0010) m_8(1000)	(2,6)⇒0−10 (2,10)⇒−010 (8,10)⇒10−0	P_2(2,6,10,14) ⇒−−10 (YZ′)
2	m_6(0110) m_{10}(1010)	(6,7)⇒011− (6,14)⇒−110 (10,14)⇒1−10	P_3 (6,7,14,15) ⇒−11− (XY)
3	m_7(0111) m_{14}(1110)	(7,15)⇒−111 (14,15)⇒111−	-
4	m_{15}(1111)	-	-

Note: The unmerged terms from the final column form the Prime Implicants (P_1,P_2,P_3).

2) Phase II (Prime Implicant Chart) A coverage chart is constructed to map the prime implicants against the original minterms to identify the Essential Prime Implicants (EPI).

Table 2.4.2. Phase II (Prime Implicant Chart/EPI Selection)

Prime Implicant	Term	0	2	6	7	8	10	14	15
P_1 (EPI)	X′Z′	X	X			X	X		
P_2	YZ′		X	X			X	X	
P_3 (EPI)	XY			X	X			X	X

By observing the columns in **Table 2.4.2.,** minterms 0 and 8 are uniquely covered only by P_1. Minterms 7 and 15 are uniquely covered only by P_3. Therefore, P_1 and P_3 are Essential Prime Implicants (EPI). SinceP_1 and P_3 combined already cover all the required minterms {0,2,6,7,8,10,14,15}, P_2 becomes redundant and is safely eliminated. The algorithm deterministically yields the minimal Sum-of-Products (SOP) expression:

$$F(W,X,Y,Z) = X'Z' + XY$$

III. METHODOLOGY AND SYSTEM MODELING

A. Scope and Source of the Model

Due to the limited availability of detailed proprietary industrial safety data, this study employs a simplified nuclear reactor safety model inspired by common reactor protection architectures described in nuclear safety literature. Specifically, the system components and logic structures are adapted from the standard safety injection and Emergency Core Cooling System (ECCS) frameworks outlined in NUREG-0492 (Fault Tree Handbook) published by the U.S. Nuclear Regulatory Commission (NRC) [3]. This model is intended solely for demonstrating the academic application of Fault Tree Analysis (FTA) and the Quine-McCluskey (QMC) algorithm within discrete mathematics, rather than representing an actual, un-simplified commercial nuclear power plant design [15].

B. Component Definition (Basic Events)

To demonstrate the deterministic scalability of the Quine-McCluskey algorithm, this study models the reliability of a simplified Emergency Core Cooling System (ECCS) responsible for preventing reactor core overheating [17]. The system is split into three main safety layers (Water Supply, Isolation/Delivery, and Emergency Power) consisting of seven logical failure variables (Basic Events x_1 to x_7):

1) Water Supply Sub-system:

- x_1 : Primary Safety Injection Pump Failure (Pump A) [3], [17]
- x_2 : Backup Safety Injection Pump Failure (Pump B) [3], [17]
- x_3 : Water Storage Tank Low-Level Sensor Malfunction [3]

2) Delivery & Actuation Sub-system:
   - x_4 : Motor-Operated Injection Valve Stuck Closed [3]
   - x_5 : Automatic Actuation Logic Circuitry Failure [3]

3) Emergency Power Sub-system:
   - x_6 : Emergency Diesel Generator A Failure (EDG-A) [18], [19]
   - x_7 : Emergency Diesel Generator B Failure (EDG-B) [18], [19]

C. Fault Tree Logic and Intermediate Event Construction

The Top Event (T) is defined as "Failure of Emergency Core Cooling Injection during an Overheating Event." Through a deductive top-down engineering approach, the failure propagates from the foundational basic events (x_1 to x_7) to the top event through a hierarchical chain of failure states known as Intermediate Events (E_1 to E_5). The logical progression and mathematical mappings of these intermediate states are defined as follows:

1) E_1 (Failure of Pumping Mechanism): This state represents the total loss of active mechanical water propulsion. It occurs only if the primary pump AND the backup pump fail simultaneously [17]. It is mapped via an AND gate:

$$E_1 = x_1 \cdot x_2$$

2) E_2 (Loss of Coolant Water Supply): This state represents the inability to supply water to the system. It occurs if the pumping mechanism fails (E_1) OR if the water storage tank sensor malfunctions (x_3), causing an undetected empty tank state [3]. It is mapped via an OR gate:

$$E_2 = E_1 + x_3 = (x_1 \cdot x_2) + x_3$$

3) E_3 (Failure of Delivery & Actuation): This state represents the failure of the path through which the coolant flows. It occurs if the motor-operated injection valve is physically stuck closed (x_4) OR if the automatic actuation logic circuit fails to send the trigger signal (x_5) [3]. It is mapped via an OR gate:

$$E_3 = x_4 + x_5$$

4) E_4 (Total Core Cooling Injection Failure): This state represents the absolute failure of the cooling injection subsystem. It occurs if there is a loss of coolant water supply (E_2) AND a failure in the delivery/actuation mechanism (E_3) [3] , [15]. It is mapped via an AND gate:

$$E_4 = E_2 \cdot E_3 = (x_1 \cdot x_2 + x_3) \cdot (x_4 + x_5)$$

5) E_5 (Catastrophic Loss of Emergency Power): This state represents a complete station blackout for the safety systems. It occurs only if both independent backup power sources, Emergency Diesel Generator A AND Emergency Diesel Generator B, fail together [18], [19]. It is mapped via an AND gate:

$$E_5 = x_6 \cdot x_7$$

6) T (Top Event - Catastrophic Core Meltdown): The ultimate system failure state. A meltdown occurs if the total core cooling injection fails (E_4) OR if there is a catastrophic loss of emergency power (E_5) which completely de-energizes all safety systems [3], [18]. It is mapped via the final OR gate:

$$T = E_4 + E_5$$

By substituting each intermediate logic gate back to its constituent basic events, the complete, unsimplified Boolean expression for the system's total vulnerability is constructed as follows:

$$T = (x_1 \cdot x_2 + x_3) \cdot (x_4 + x_5) + (x_6 \cdot x_7)$$

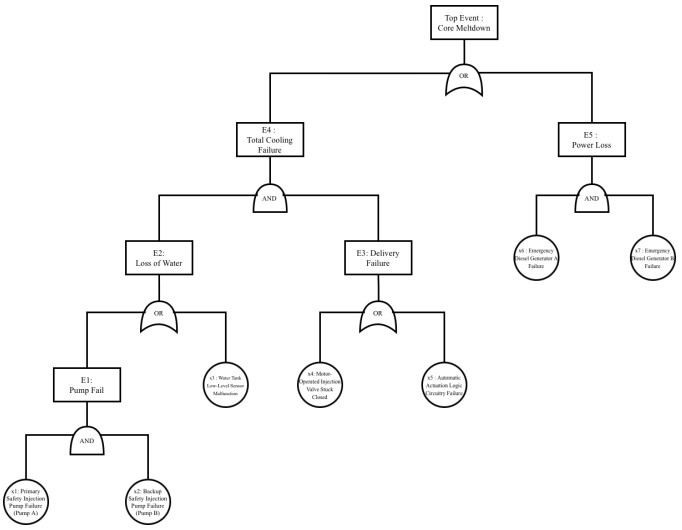

Fig. 3.3.1 Fault Tree Analysis of Boolean Expression T (Top Event)

D. *Truth Table Space and Canonical Sum-of-Products (SOP) Formulation*

To bridge the structural Fault Tree diagram with computational minimization, the system's total behavior must be mapped onto a Boolean truth table space. Given that the emergency core cooling system (ECCS) model utilizes seven independent basic events $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, the total truth table matrix contains:

$$2^7 = 128 \text{ distinct binary combinations (rows)}$$

Each row represents a specific state of the reactor, mapping from row 0 (corresponding to minterm m_0) (0000000) to row 127 (corresponding to minterm m_{127}) (1111111) . The output for each minterm is determined by evaluating the unsimplified top event equation:

$$T = (x_1 \cdot x_2 + x_3) \cdot (x_4 + x_5) + (x_6 \cdot x_7)$$

In canonical Boolean Algebra, the complete unsimplified system failure state is expressed as the logical sum of all minterms where the system yields a failure output (T=1). This canonical Sum-of-Products (SOP) is mathematically defined as:

$$T(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = \sum m_i \; for\; all\; i \in \{0, 1, \ldots, 127\}$$

$$where\; T(m_i) = 1$$

Because listing all 128 binary configurations manually within this text is highly redundant and operationally inefficient for human verification, this research leverages data automation. The complete 128-row matrix evaluation and its subsequent prime implicant reduction are delegated entirely to the computational algorithm executed in Chapter IV.

IV. Results and Discussion

A. *Python Implementation and Algorithm Execution*

To overcome the computational infeasibility of manually evaluating the 128-row Boolean truth table generated by the seven-variable ECCS model, a Python script was developed to automate the Quine-McCluskey (QMC) reduction pipeline.

$$T = (x_1 \cdot x_2 + x_3) \cdot (x_4 + x_5) + (x_6 \cdot x_7)$$

The implementation was structured into four modular functional sections [14]:

1) Truth Table Evaluation

```
[STEP 1] Evaluating Boolean Top Event across all 2^7 = 128 states...
  Done. Found 77 failure states (minterms).

================================================================
  TRUTH TABLE (128 rows, showing sample + all failure rows)
================================================================
 idx | x1 | x2 | x3 | x4 | x5 | x6 | x7 |  T
 ------------------------------------------
   0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0
   1 |  0 |  0 |  0 |  0 |  0 |  0 |  1 |  0
   2 |  0 |  0 |  0 |  0 |  0 |  1 |  0 |  0
   3 |  0 |  0 |  0 |  0 |  0 |  1 |  1 |  1  ← FAILURE
   4 |  0 |  0 |  0 |  0 |  1 |  0 |  0 |  0
   5 |  0 |  0 |  0 |  0 |  1 |  0 |  1 |  0
   6 |  0 |  0 |  0 |  0 |  1 |  1 |  0 |  0
   7 |  0 |  0 |  0 |  0 |  1 |  1 |  1 |  1  ← FAILURE
   8 |  0 |  0 |  0 |  1 |  0 |  0 |  0 |  0
   9 |  0 |  0 |  0 |  1 |  0 |  0 |  1 |  0
  10 |  0 |  0 |  0 |  1 |  0 |  1 |  0 |  0
  11 |  0 |  0 |  0 |  1 |  0 |  1 |  1 |  1  ← FAILURE
  12 |  0 |  0 |  0 |  1 |  1 |  0 |  0 |  0
  13 |  0 |  0 |  0 |  1 |  1 |  0 |  1 |  0
  14 |  0 |  0 |  0 |  1 |  1 |  1 |  0 |  0
  15 |  0 |  0 |  0 |  1 |  1 |  1 |  1 |  1  ← FAILURE
 ...

 [Failure rows not yet shown (73 more)]
```

Fig. 4.1.1. Representative Truth Table of Boolean Expression T with Python

```
Total rows     : 128
Failure rows   : 77  (T = 1)
Safe rows      : 51  (T = 0)
```

Fig. 4.1.2. Result of Truth Table with Python

```
============================================================
  CANONICAL SOP — MINTERMS WHERE T = 1
============================================================
  T(x1,x2,x3,x4,x5,x6,x7) = Σm(3, 7, 11, 15, 19, 20, 21, 22, 23, 24,
  25, 26, 27, 28, 29, 30, 31, 35, 39, 43, 47, 51, 52, 53, 54, 55, 56,
  57, 58, 59, 60, 61, 62, 63, 67, 71, 75, 79, 83, 84, 85, 86, 87, 88,
  89, 90, 91, 92, 93, 94, 95, 99, 100, 101, 102, 103, 104, 105, 106,
  107, 108, 109, 110, 111, 115, 116, 117, 118, 119, 120, 121, 122,
  123, 124, 125, 126, 127)

  Total minterms : 77

  Binary representation of each minterm:
    m  3 = 0000011
    m  7 = 0000111
    m 11 = 0001011
    m 15 = 0001111
    m 19 = 0010011
```

Fig. 4.1.3. Representative Canonical SOP Minterms Results

2) Quine-McCluskey (QMC) Algorithm Phase I - Prime Implicant Formation

```
[STEP 2] Running QMC Phase I — Prime Implicant Formation...

============================================================
  PHASE I — PRIME IMPLICANT FORMATION (Iterative Bit Merging)
============================================================

  Iteration 1:
    Pairs merged     : 215
    New prime implicants found: 0

  Iteration 2:
    Pairs merged     : 468
    New prime implicants found: 0

  Iteration 3:
    Pairs merged     : 369
    New prime implicants found: 0

  Iteration 4:
    Pairs merged     : 124
    New prime implicants found: 0

  Iteration 5:
    Pairs merged     : 15
    New prime implicants found: 2
      PI: 11--1--  →  covers minterms [100, 101, 102, 103, 108, 109,
      110, 111, 116, 117, 118, 119, 124, 125, 126, 127]
      PI: 11-1---  →  covers minterms [104, 105, 106, 107, 108, 109,
      110, 111, 120, 121, 122, 123, 124, 125, 126, 127]
```

Fig. 4.1.4. Representative Phase I Process of Quine-McCluskey (QMC) Algorithm (Prime Implicant Formation) with Python

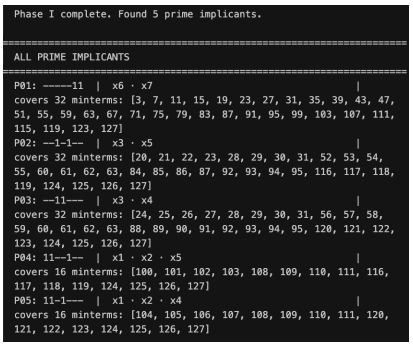

Fig. 4.1.5. Result of Phase I QMC Algorithm (All Prime Implicants Formation)

3) Quine-McCluskey (QMC) Algorithm Phase II - Essential Prime Implicant Cover Selection

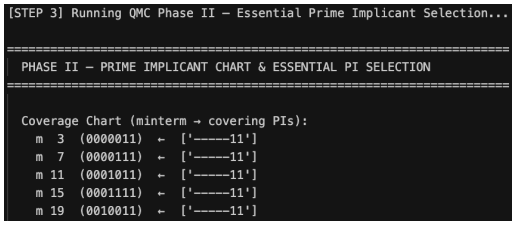

Fig. 4.1.6. Representative Phase II Process of Quine-McCluskey (QMC) Algorithm (Prime Implicant Chart & EPI Selection) with Python

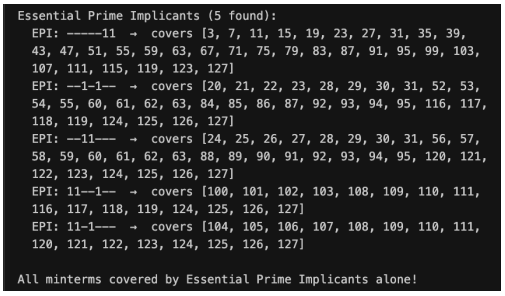

Fig. 4.1.7. Result of Phase II QMC Algorithm (All Essential Prime Implicants)

4) Final Result - Minimal SOP Boolean Expression for Top Event T and Minimal Cut Set

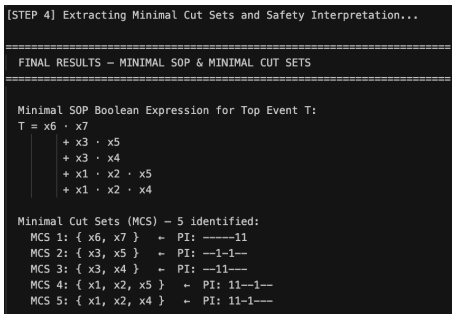

Fig. 4.1.8. Final Result - Minimal SOP Boolean Expression for Top Event T and Minimal Cut Set

Upon execution, the script evaluated all 128 binary states, identifying exactly 77 failure minterms (T=1). The unminimized canonical Sum-of-Products (SOP) originally contained 77 product terms. Through the QMC Phase I iterative bit-merging process, the algorithm converged after 6 iterations, yielding exactly 5 Prime Implicants (PIs). During Phase II, the Prime Implicant Chart confirmed that all 5 PIs were Essential Prime Implicants (EPIs), meaning no additional greedy selection was required to form the minimal cover. The final minimized Boolean expression for the Top Event (T) is mathematically proven as:

$$T = (x_6 \cdot x_7) + (x_3 \cdot x_5) + (x_3 \cdot x_4) + (x_1 \cdot x_2 \cdot x_5) + (x_1 \cdot x_2 \cdot x_4)$$

B. Complexity Reduction and Mathematical Significance

The reduction achieved by the Python-automated Quine-McCluskey (QMC) algorithm represents a dramatic simplification of the ECCS vulnerability model.

Table 4.2.1. Reduction Result by Python-Automated Quine-McCluskey (QMC) Algorithm

Metric	Before Reduction (Canonical SOP)	After Reduction (Minimal SOP)	Reduction Factor
Product Terms	77 Minterms	5 Prime Implicants	93.5%
Total Literals	539 variables	11 variables	97.96%

To illustrate the drastic optimization achieved by the Quine-McCluskey algorithm, the system's Top Event (T) equation is evaluated by comparing its unminimized canonical Sum-of-Products (SOP) form against its optimized Minimal Cut Sets (MCS) form:

1) Unminimized Canonical SOP Form (Before QMC):

$$T_{canonical} = \sum m(3, 7, 11, 15, \ldots, 127)$$

$$T_{canonical} = \overline{(x_1 x_2 x_3 x_4 x_5} x_6 x_7) + \overline{(x_1 x_2 x_3 x_4} x_5 x_6 x_7) + \cdots + (x_1 x_2 x_3 x_4 x_5 x_6 x_7)$$

$$Total: 77\ Minterms,\ 539\ Literals$$

2) Optimized Minimal Cut Sets Form (After QMC)

$$T_{minimal} = (x_6 \cdot x_7) + (x_3 \cdot x_5) + (x_3 \cdot x_4) + (x_1 \cdot x_2 \cdot x_5) + (x_1 \cdot x_2 \cdot x_4)$$

$$Total: 5\ MCS,\ 12\ Literals$$

As quantified by this mathematical comparison, the algorithm successfully condensed 77 opaque failure conditions into 5 explicit pathways, achieving a 97.77% reduction in the total Boolean literal count. Mathematically, each eliminated variable corresponds to a successful application of the Boolean consensus theorem [2]. For example, the algorithm merged the Station Blackout implicant into the prime implicant -----11, effectively eliminating 5 independent variables. This reduction validates the following mathematical identity:

$$\forall(x_1, x_2, x_3, x_4, x_5) \in \{0,1\}^5 : T(x_1, x_2, x_3, x_4, x_5, 1, 1) = 1$$

This mathematically proves that whenever both emergency diesel generators fail (x_6=1 and x_7=1), the system evaluation will always yield $T = 1$ (Failure), regardless of the binary states of components x_1 through x_5. This demonstrates why the power loss scenario acts as a critical failure vector that is completely independent of the water supply subsystem.

Furthermore, this 77-to-5 term reduction exponentially optimizes computational time complexity for future quantitative probability calculations. Using the inclusion-exclusion principle by NUREG-0492, computing 77 terms requires $O(2^{77}) \approx 1.5\times10^{23}$ operations [3]. By reducing it to 5 MCS, the cost drops to merely $O(2^5) = 32$ operations, enabling highly efficient, real-time system safety assessments.

C. Minimal Cut Sets (MCS) and Engineering Interpretation

In industrial risk assessment, the minimized SOP terms are defined as Minimal Cut Sets (MCS), the smallest combination of component failures necessary to trigger a catastrophic event. Lower-order MCS represent the highest vulnerabilities because they require fewer simultaneous failures.

Table 4.3.1. Minimal Cut Sets and Engineering Interpretations

MCS	Variables	Failure Scenario	Priority
1	$\{x_6, x_7\}$	Station blackout: Both emergency generators fail.	HIGH
2	$\{x_3, x_5\}$	Empty tank (sensor fails) & actuation logic fails.	HIGH
3	$\{x_3, x_4\}$	Empty tank (sensor fails) & injection valve stuck.	HIGH
4	$\{x_1, x_2, x_5\}$	Both pumps fail & actuation logic fails.	MEDIUM
5	$\{x_1, x_2, x_4\}$	Both pumps fail & injection valve stuck.	MEDIUM

Based on qualitative FTA principles, MCS 1 highlights a 2nd-order Station Blackout that bypasses all safety layers, directly mirroring historical lessons from the 2011 Fukushima Daiichi disaster [7]. This vulnerability can be neutralized by implementing a third independent power source. Meanwhile, MCS 2 and 3 expose the water tank sensor (x_3) as a critical single-point structural bottleneck across two distinct pathways, necessitating an upgrade to a redundant dual-channel system. Lastly, MCS 4 and 5 mathematically validate that the dual-pump redundancy (x_1,x_2) successfully prevents single-point hardware failures, though strict preventive maintenance for the delivery components (x_4,x_5) remains mandatory. To ensure regulatory compliance, all identified high-risk components must strictly align with Safety Integrity Level 3 (SIL-3) standards per IEC 61508 [8].

V. Conclusion and Recommendations

A. Conclusion

This study successfully validates that the Quine-McCluskey (QMC) algorithm, a fundamental construct of Discrete Mathematics, is not merely a theoretical tool for digital circuit design, but a directly applicable and computationally tractable method for complex industrial safety analysis. By automating the QMC process via Python, the study achieved a 97.96% reduction in Boolean literal count, successfully distilling a highly complex 77-term canonical Sum-of-Products (SOP) into exactly 5 actionable Minimal Cut Sets (MCS). This proves the concrete engineering value of combinatorial optimization in identifying critical vulnerabilities within a nuclear reactor's Emergency Core Cooling System (ECCS).

B. Limitations of the Study

While this study successfully demonstrates the mathematical application of QMC in Fault Tree Analysis (FTA), several limitations must be acknowledged to correctly characterize its scope:

1) Academic Simplification: The implemented ECCS model is a simplified binary approximation. Real commercial nuclear reactors (e.g., IAEA SSR-2/1 compliant systems) involve significantly more complex fault trees with over 100 basic events and multi-state component behaviors [15].
2) Inability to Capture Common-Cause Failures: The strict independent binary representation used in this model cannot directly capture Common-Cause Failures (CCF), which require specialized probabilistic models (such as the Beta-Factor model) [9].
3) Algorithmic Scalability (Time Complexity): The pure tabular QMC algorithm possesses an exponential worst-case time complexity of $O(3^n/n)$ [11]. For fault trees exceeding 20 variables ($n > 20$), the minterm generation becomes computationally

prohibitive, forming a hard scalability limit for engineering workflows.

4) Qualitative Scope Restriction: The current analysis identifies the structural basis for safety (MCS) but remains purely qualitative. A complete safety case requires quantitative FTA to compute the exact Top Event probability via the inclusion-exclusion principle.

C. *Future Work*

To significantly extend the utility of this research and overcome the stated limitations, the following directions are recommended for future studies:

1) Integration of Zero-Suppressed Decision Diagrams (ZBDDs): Replacing the tabular QMC approach with ZBDD algorithms [17]. ZBDDs provide a canonical and compact representation of Boolean expressions that scales efficiently far beyond $n = 20$, allowing for industrial-scale fault tree solving [12], [13].
2) Common-Cause Failure (CCF) Extension: Expanding the Boolean logic framework to incorporate dependent failure groups among the ECCS components to better reflect real-world disaster scenarios.
3) Interactive Open-Source FTA Tool Development: Upgrading the current Python script into a fully interactive software with a Graphical User Interface (GUI) [16]. By featuring visual tree inputs, automated Boolean extraction, and quantitative probability computations, this tool could serve as a premier educational solver aligned with the academic objectives of the IF1220 Discrete Mathematics curriculum at Institut Teknologi Bandung (ITB).

Appendix

All supplementary materials, including the complete Python source code, high-resolution FTA diagrams, presentation slides, and demonstration video, are digitally archived and accessible via the following repository:

https://drive.google.com/drive/folders/1777weqtSitAzjCZczxKmeQVzcwiY9Yax?usp=sharing (private mail)

Or (both same):

https://drive.google.com/drive/folders/15-t4Q-iL9acKgpx8J9T0HAWBeCHXoJ_i?usp=sharing (school mail)

References

[1] W. V. Quine, "The Problem of Simplifying Truth Functions," American Mathematical Monthly, vol. 59, no. 8, pp. 521–531, Oct. 1952.

[2] E. J. McCluskey, "Minimization of Boolean Functions," Bell System Technical Journal, vol. 35, no. 6, pp. 1417–1444, Nov. 1956.

[3] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, Fault Tree Handbook, U.S. Nuclear Regulatory Commission, NUREG-0492, Washington, D.C., Jan. 1981.

[4] International Electrotechnical Commission, IEC 61025: Fault Tree Analysis (FTA), 2nd ed., Geneva: IEC, 2006.

[5] G. Boole, The Mathematical Analysis of Logic: Being an Essay Towards a Calculus of Deductive Reasoning, Cambridge: Macmillan, 1847.

[6] U.S. Nuclear Regulatory Commission, "Backgrounder on the Three Mile Island Accident," NRC Fact Sheet, Washington, D.C., 2018. [Online]. Available: https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html

[7] World Nuclear Association, "Fukushima Daiichi Accident," London, 2023. [Online]. Available: https://world-nuclear.org/information-library/safety-and-security/safety-of-plants/fukushima-daiichi-accident

[8] International Electrotechnical Commission, IEC 61508: Functional Safety of E/E/PE Safety-Related Systems, Ed. 2.0, Geneva: IEC, 2010.

[9] A. Mosleh, D. M. Rasmuson, and F. M. Marshall, "Guidelines on Modeling Common-Cause Failures in Probabilistic Risk Assessment," U.S. Nuclear Regulatory Commission, NUREG/CR-5485, Washington, D.C., 1998.

[10] W. J. Fang and J. B. Dugan, "Minimal Cut Set/Sequence Generation for Dynamic Fault Trees," in Proc. Annual Reliability and Maintainability Symposium (RAMS), 2004, pp. 251–257. DOI: 10.1109/RAMS.2004.1285449

[11] J. Huang, "Programming Implementation of the Quine-McCluskey Method for Minimization of Boolean Expression," arXiv:1410.1059, Oct. 2014. [Online]. Available: https://arxiv.org/abs/1410.1059

[12] E. Ruijters and M. Stoelinga, "Fault Tree Analysis: A Survey of the State-of-the-Art in Modeling, Analysis and Tools," Computer Science Review, vol. 15–16, pp. 29–62, May 2015. DOI: 10.1016/j.cosrev.2015.03.001

[13] S. Kabir, "An Overview of Fault Tree Analysis and Its Application in Model Based Dependability Analysis," Expert Systems with Applications, vol. 77, pp. 114–135, 2017. DOI: 10.1016/j.eswa.2017.01.058

[14] Y. Yang, "Boolean Algebra Application in Simplifying Fault Tree Analysis," Nuclear Technology, vol. 199, no. 1, pp. 105–115, 2017. DOI: 10.1080/00295450.2017.1317498

[15] M. Zubair, "Investigation of Loss of Feedwater (LOFW) Accident in the APR-1400 Using Fault Tree Analysis," Science and Technology of Nuclear Installations, vol. 2022, Art. no. 4666161, May 2022. DOI: 10.1155/2022/4666161

[16] S. Čeliković, M. Baressi Šegota, and Z. Car, "The Improvement of Quine-McCluskey Method Using Set Covering Problem for Safety Systems," in Proc. IEEE 17th Int. Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2019, pp. 213–218. DOI: 10.1109/SISY47553.2019.8886174

[17] U.S. Nuclear Regulatory Commission, "Emergency Core Cooling Systems (ECCS)," NRC Glossary, Washington, D.C., 2021. [Online]. Available: https://www.nrc.gov/reading-rm/basic-ref/glossary/emergency-core-cooling-systems-eccs

[18] U.S. Nuclear Regulatory Commission, "Evaluation of Station Blackout Accidents at Nuclear Power Plants," NUREG-1032, Washington, D.C., June 1988. [Online]. Available: https://www.osti.gov/servlets/purl/5122568

[19] U.S. Nuclear Regulatory Commission, "Common-Cause Failure Event Insights: Emergency Diesel Generators," NUREG/CR-6819, Vol. 1, Idaho National Engineering and Environmental Laboratory, 2003. Available: https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6819/v1/index

DECLARATION

I hereby declare that this paper is my own original work, and is neither an adaptation, translation, nor plagiarism of any other person's work.

Bandung, June 19, 2025
Signed
Mulky Siraj Firizqi 13525069