CHAPTER 1

INTRODUCTION

1.1 GENERAL

Digital systems, even when designed with highly reliable components, do not operate forever without developing faults. When a system ultimately develops a fault, it has to be detected and located to remove its effect. Fault detection means the discovery of something wrong in a digital system or circuit. Fault location means the identification of the faults with components, functional modules or subsystems, depending on the requirements. Fault diagnosis includes both fault detection and fault location.

Fault detection in a logic circuit is carried out by applying a sequence of test inputs and observing the resulting outputs. Therefore, the cost of testing includes the generation of test sequences and their application. One of the main objectives in testing is to minimize the length of the test sequence. Any fault in a non-redundant n-input combinational circuit can be completely tested by applying all \(2^n\) input combinations to it. However, \(2^n\) increases rapidly as \(n\) increases. Even for \(n=60\), at a rate of 10,000 tests per second, the total test time for the circuit would be about 3.5 million years (Parag Lala 1990). A complete truth–table exercise of a logic circuit is not necessary. Only the number of input combinations which detect most of the faults in the circuit is necessary.
In order to determine the faults that have been detected by a set of test patterns, a process known as “fault simulation” is performed. Fault simulation is a process of applying every test pattern to the fault-free circuit and comparing the results of the fault simulated circuits. Faults can be simulated one at a time or by the method of parallel simulation or by deductive simulation (requires more memory). The time required to compute the test patterns for a combinational circuit grows in proportion to the square of the number of gates in the circuit. Hence, the computation time required for test generation will be prohibitive for circuits of VLSI complexity. Several methods are available for deriving test sets for combinational circuits.

Logic circuits for digital systems may be combinational or sequential. A combinational circuit consists of logic gates whose output at any time is determined directly from the present combination of inputs without regard to the previous inputs. A combinational circuit performs a specific information processing operation fully specified logically by a set of Boolean functions.

Logic networks are usually designed by using ‘AND’ and ‘OR’ gates. However, the networks using AND-EXOR gates have some advantages over the conventional AND-OR networks. First, such networks often require fewer gates and interconnections than those designed using AND and OR gates. Examples of such networks include arithmetic, telecommunication and error correcting circuits. Second, they can be made easily testable.

The fault detection problem of combinational logic circuits is of extreme importance. A number of methods such as D-algorithm, Fan algorithm and testable design are available. In general, any arbitrary logic function can be expressed in Reed-Muller Canonical (RMC) form as,
F = \bigoplus a_0 a_1 x_1^* \bigoplus a_2 x_2^* \bigoplus \ldots \bigoplus a_n x_n^* \bigoplus a_{n+1} x_1^* x_2^* \bigoplus \ldots \bigoplus a_m x_1^* x_2^* \ldots x_n^*

where, \( x_n^* \) can be \( x_n \) the positive polarity or its complement, \( a_n \) is either 0 or 1 and \( m = 2^n - 1 \). For such a logic function there exists four forms, namely, Positive Polarity Reed-Muller (PPRM), Fixed Polarity Read-Muller (FPRM), Generalized Reed-Muller (GRM) and Exclusive-OR Sum of Products (ESOP).

**PPRM**

The expression shown in the previous paragraph is called Positive Polarity Reed-Muller (PPRM), if only positive polarities are allowed for each input variable. The coefficients \( a_0, a_1, a_2, \ldots \) are unique for a given function. For example, \( x_1 x_2 \oplus x_2 x_3 \) is a PPRM expression since all the variables appear with only positive polarities.

**FPRM**

If either a positive literal \( x_i \) only or negative literal \( x_i^* \) only is used for a particular variable, it is called a Fixed Polarity Reed-Muller expression (FPRM). For example, \( x_1 x_2 \oplus x_2 x_3 \) belongs to this class.

**GRM**

If an AND–EXOR expression has no restrictions on the polarities of variables, but does not allow the same set of variables to appear in more than one product term, it is called Generalized Reed–Muller expression (GRM). For example, \( x_1 x_2^* \oplus x_2 x_3^* \) is a GRM.
ESOP

An expression consisting of arbitrary product terms combined by EXOR operations is called an EXOR Sum-of-Products expression (ESOP). There is no restriction on the variables. For example, \(x_1x_2x_3 \oplus x_1'x_2x_3\) is an ESOP, but not a GRM, as the same set of variables \(\{x_1, x_2, x_3\}\) appears in two product terms. The following relation holds PPRM \(\subset\) FPRM \(\subset\) GRM \(\subset\) ESOP among the four forms.

![Diagram](image)

**Figure 1.1 Relation among various classes of AND-EXOR expression**

ESOP is the most general form and in most cases gives a significantly small number of product terms among the four forms. Table 1.1 shows the number of product terms required to realize some arithmetic functions for different forms. It has been observed that ESOPs usually require fewer product terms than GRMs; PPRMs often need more product terms than GRMs. For many Boolean functions, GRMs require fewer product terms than the classical AND–OR expressions.
Due to the total freedom of input polarity and product term selection, the minimum number of product terms required to represent an arbitrary function in ESOP form can never be larger than the minimum number of product terms in any of the canonical Reed-Muller forms. Thus, most of the circuits are realized using ESOP forms.

Main contributions described in the present project are a highly testable ESOP realization and a minimal universal test set that detects all possible single stuck-at, double stuck-at, AND-bridging and OR-bridging faults in the entire circuit, including the faults in the primary input and output leads.

### Table 1.1 Number of product terms for certain arithmetic functions of various RMC forms

<table>
<thead>
<tr>
<th>Data</th>
<th>PPRM</th>
<th>FPRM</th>
<th>GRM</th>
<th>ESOP</th>
<th>SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adr4</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>31</td>
<td>75</td>
</tr>
<tr>
<td>Inc8</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>37</td>
</tr>
<tr>
<td>Log8</td>
<td>253</td>
<td>193</td>
<td>105</td>
<td>96</td>
<td>123</td>
</tr>
<tr>
<td>mlp4</td>
<td>97</td>
<td>97</td>
<td>71</td>
<td>61</td>
<td>121</td>
</tr>
<tr>
<td>nrm4</td>
<td>216</td>
<td>185</td>
<td>96</td>
<td>69</td>
<td>120</td>
</tr>
<tr>
<td>Rdm8</td>
<td>56</td>
<td>56</td>
<td>31</td>
<td>31</td>
<td>76</td>
</tr>
<tr>
<td>Rot8</td>
<td>225</td>
<td>118</td>
<td>51</td>
<td>35</td>
<td>57</td>
</tr>
<tr>
<td>Sqr8</td>
<td>168</td>
<td>168</td>
<td>121</td>
<td>112</td>
<td>178</td>
</tr>
<tr>
<td>Sym9</td>
<td>210</td>
<td>173</td>
<td>126</td>
<td>51</td>
<td>84</td>
</tr>
<tr>
<td>Wgt8</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>58</td>
<td>255</td>
</tr>
</tbody>
</table>

### 1.2 FAULTS IN DIGITAL CIRCUITS

#### 1.2.1 Failures and faults

A fault is a physical defect which may or may not cause a failure. A fault is characterized by its nature, value, extent and duration. The nature of a
fault can be classified as logical and non-logical. A logical fault causes the logical value at a point in a circuit to become opposite to the specified value. Non-logical faults include the rest of the faults such as the malfunction of the clock signal and power failure. The value of a logical fault at a point in the circuit indicates whether the fault creates a fixed or varying erroneous logical value. The extent of a fault specifies whether the effect of a fault is localized or distributed. A local fault affects only a single variable whereas a distributed fault affects more than one. A logical fault, for example, is a local fault while the malfunction of the clock is a distributed fault. The duration of the fault refers to whether the fault is permanent or temporary.

1.2.2 Modeling of Faults

Faults in a circuit may occur due to defective components, breaks in signal lines, lines shorted to ground or power supply, short circuiting of signal lines, excessive delays and design rule violations. Poor designs may also result in hazards, races or metastable flip-flop behavior in a circuit. Such faults manifest themselves as intermittents throughout the life of the circuit. In general, the effect of a fault is represented by means of a model, which represents the change that the fault produces in circuit signals. The main fault models in use today are stuck-at fault, bridging fault and stuck-open fault.

Table 1.2 shows some of the causes of faults. The first column shows the fault level, that is, whether the fault occurs in the logic gates in the chip or in the package. The second column describes the physical fault. There are too many of these and there should be a way to reduce and simplify their effects by using a fault model (Michael John Sebastian Smith 2004).
Table 1.2 Mapping physical faults to logical faults

<table>
<thead>
<tr>
<th>Fault level</th>
<th>Physical fault</th>
<th>Degradation fault</th>
<th>Open-circuit fault</th>
<th>Short-circuit fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip</td>
<td>Leakage or short between package leads</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Broken, misaligned, or poor wire bonding</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Surface contamination, moisture</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Metal migration, stress, peeling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Metallization (open or short)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gate</td>
<td>Contact opens</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gate to S/D junction short</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Field-oxide parasitic device</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gate-oxide imperfection, spiking</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Mask misalignment</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

There are several types of fault models. First, a physical fault is mapped to a logical fault. Next, distinction is made between those logical faults that degrade the ASIC performance and those faults that are fatal and stop the ASIC from working. There are three kinds of logical faults as shown in Table 1.2, namely a degradation fault, an open-circuit fault, and a short-circuit fault (Michael John Sebastian Smith 2004).
A degradation fault may be a parametric fault or a delay fault (timing fault). A parametric fault might lead to an incorrect switching threshold in a TTL/CMOS circuits and can be tested using a production tester. A delay fault might lead to a critical path being slower than the specification. Delay faults are much harder to test in production.

An open-circuit fault results from physical faults such as a bad contact, a piece of metal that is missing or over-etched, or a break in polysilicon line.

A short-circuit fault results from such physical faults such as,

- under-etching of metal
- spiking
- pin holes or shorts across the gate oxide
- diffusion shorts

These faults result in a circuit being accidentally connected as a short circuit. Most of the short-circuit faults occur in interconnect. They are often called bridging faults. This type of fault usually results from metal coverage problems that lead to shorts. There are two basic classification of such faults, namely, feedback bridging faults and non-feedback bridging faults. In a feedback type there exists an interconnection from the output of the circuit to an input point. Bridging faults are a frequent problem in CMOS ICs.

Table 1.3 shows some of the defects available in a PCB and frequency of occurrence (Batson 1985).
<table>
<thead>
<tr>
<th>Defect class</th>
<th>Percentage probability of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shorts</td>
<td>51</td>
</tr>
<tr>
<td>Opens</td>
<td>1</td>
</tr>
<tr>
<td>Missing components</td>
<td>6</td>
</tr>
<tr>
<td>Wrong components</td>
<td>13</td>
</tr>
<tr>
<td>Reversed components</td>
<td>6</td>
</tr>
<tr>
<td>Bent leads</td>
<td>8</td>
</tr>
<tr>
<td>Analog specifications</td>
<td>5</td>
</tr>
<tr>
<td>Digital logic</td>
<td>5</td>
</tr>
<tr>
<td>Performance (timing)</td>
<td>5</td>
</tr>
</tbody>
</table>

### 1.3 CLASSIFICATION OF FAULTS

Faults in general, can be broadly classified as follows, namely,

- Single and multiple stuck-at faults
- Transistor open and short faults
- Memory faults
- Functional faults (processors)
- Delay faults (transition, path)
- Bridging faults
- PLA faults (stuck-at, cross-point, bridging)
1.3.1 Stuck-at Faults

The most common model used for logical faults is the single stuck-at fault. It assumes that a fault in a logic gate results in one of its inputs or output being fixed to either logic 0 (stuck-at-0) or logic 1 (stuck-at-1). Stuck-at-0 and stuck-at-1 faults are often abbreviated as s-a-0 and s-a-1 respectively. The stuck-at fault model, often referred to as the classical fault model, offers a good representation for the most common types of failures, e.g. short-circuits and open-circuits in many technologies.

Figure 1.2 illustrates the Transistor-Transistor Logic (TTL) realization of a NAND gate. The numbers 1, 2 and 3 indicate the places where opens may principally occur, while 4 and 5 indicate the basic types of shorts.

Figure 1.2 Schematic diagram of a NAND gate
**Signal line open**

This fault prevents the sink current $I_s$ from flowing through the emitter of the input transistor $T_1$ into the output of the preceding gate. Thus, the input appears to be connected to a constant level 1, i.e. s-a-1.

**Supply voltage open**

In this case, the gate is deprived of its supply voltage and thus neither the current $I_s$, which would switch the transistor $T_1$ on, nor the current $I_T$, which may excite $T_3$, can flow. Both the output transistors are cutoff and the output appears to be open. The fault can be interpreted as the gate output s-a-1 for the next stage.

**Ground open**

This fault prevents transistors $T_2$ and $T_4$ from conducting and hence the current $I_T$ continually switches transistor $T_3$ on. The output has the value of a normal logic 1, i.e the fault may be interpreted as output s-a-1.

**Signal line and $V_{ce}$ short-circuited**

This fault is of the s-a-1 type, but the transistor $T_4$ of the preceding gate is overloaded. Thus, a secondary fault can be caused.

**Signal line and ground short-circuited**

The fault of this type may be interpreted as s-a-0. The “stuck-at” model is also used to represent multiple faults in circuits. In a “multiple stuck-at fault” it is assumed that more than one signal line in the circuit are stuck-at logic 1 or logic 0. In other words, a group of stuck-at faults exist in the circuit at the same time.
The stuck-at model has gained wide acceptance in the past because of its relative success with small scale integration. However, faults in the MOS circuits do not necessarily produce logical faults (they involve modification of the network function) that can be described as stuck-at faults.

1.3.1.1 Single stuck-at fault

A fault is called as a single stuck-at fault if it involves only one line of the circuit. The basic assumptions that characterize the single stuck-at fault model are,

- The faulty line is permanently set either 0 or 1.
- Fault can be at an input or output of a gate.

Assume a single stuck-at fault model, which allows only one stuck-at fault in the entire circuit. A n-input AND gate requires (n+1) test vectors to detect a single stuck-at fault in its input or output. The tests for 3 input AND gate for example would be \{111, 011, 101, 110\}. In this test set \{111\} detects a s-a-0 fault on any of the inputs or the output; and the remaining tests, commonly referred to as walking-0 tests, detects an s-a-1 on any of the inputs and the output of the gate. The single stuck-at fault model is the most widely used, due to its simplicity in terms of employing it in CAD and test tools. The stuck-at fault is used at both the chip and PCB levels. Figures 1.3 and 1.4 represent the single stuck-at fault model in the input and output lines of the gate circuit.

![Figure 1.3 AND gate with a single stuck-at-0 fault](image-url)
**Figure 1.4 Faulty inverter with a single stuck-at-1 fault**

In Figure 1.2, one of the gate inputs is stuck-at-0. Therefore, irrespective of what is applied to the inputs, the output is always one. In Figure 1.3, the output of the inverter is stuck-at-1. Hence, no matter what is applied to the input, the output will be logic 1. Although the stuck-at fault model is popular, it is not a realistic model for any kind of circuit defects. However, test vectors which are developed to detect stuck-at faults also detect a variety of other faults when the coverage of stuck-at faults is significantly high.

**Figure 1.5 Circuit without stuck-at fault**

**Figure 1.6 Circuit with stuck-at -1 fault at point ‘a’**
The stuck-at faults may cause the logic function of a circuit to get changed as illustrated in Figures 1.5 to 1.7. When there is no fault in the circuit it can be observed that the output is \( Z = X_1 X_3 + X_3 X_4 \) (Figure 1.5). If there is a fault (s-a-1) at location ‘a’ in the circuit, the output would be as \( Z = X_1 + X_2 X_3 \) (Figure 1.6) or for a fault (s-a-1) at location ‘g’ the output \( Z \) is \( X_2 X_3 \) as shown in the Figure 1.7.

1.3.1.2 Multiple stuck-at faults

A fault is called as a multiple stuck-at fault, if the fault involves more than one line of the circuit simultaneously. Unlike a single stuck at fault model, a multiple fault model represents a condition caused by the presence of a group of single faults (Figure 1.8). In Figure 1.9 the stuck-at-0 at terminal In-1 and stuck-at-1 at terminal In-2 is an example of double stuck-at faults.
Figure 1.9 A NAND gate with two stuck-at faults

1.3.2 Bridging Faults

A bridging fault occurs when two leads in the logic network are connected accidently and “wired logic” is performed at the connection. Depending on whether positive or negative logic is used, the faults have the effect of ANDing or ORing the signals involved respectively as shown in the Figure 1.10. With the stuck-at faults, if there are $n$ lines in the circuit, there are $2n$ possible single stuck-at faults and $(3^n - 1)$ possible multiple stuck-at faults. With bridging faults, when the bridging between any $s$ lines in a circuit is considered, the number of single bridging faults will alone be $\binom{n}{s}$ and the number of multiple bridging faults will be very much larger (Parag Lala 1990). If there is bridging among $s$ input lines of circuit, it has an input bridging fault of multiplicity $s$. A feedback bridging fault of multiplicity $s$ results when there is bridging among the output of the circuit and $s$ input lines.
POSITIVE LOGIC

Original circuit

Equivalent Faulty circuit

(a)

NEGATIVE LOGIC

Original circuit

Equivalent Faulty circuit

(c)

Figure 1.10 Examples of bridging faults

Figures 1.11 and 1.12 show the logical models of input and feedback bridging respectively.
Figure 1.11 Logical model of input bridging

Figure 1.12 Logical model of feedback bridging

Figure 1.13 Bridging fault example
A bridging can be modeled at the gate or transistor level as a short between two or more signal lines. It can be an AND bridge or an OR bridge. In an AND bridging fault, the post fault values of the lines involved have the same value as the logical AND-ed value of their pre-fault values. Similarly, for an OR-bridging fault the post fault values of the lines involved have the same value as the logical OR-ed value of their pre-fault values. The choice depends on the relative strength of signals on the bridged nodes. Figure 1.13 shows an example of a bridging fault. The following assumptions are usually made during bridge fault.

- At least two nodes of a circuit are shorted together.
- The bridged path is usually assumed to be of low resistance.
- Three classes are typically possible, namely,
  - Bridging within a logic element (transistor gates, sources, or drains shorted together)
  - Bridging between the logic nodes (i.e., inputs or outputs of logic elements) without feedback
  - Bridging between the logic nodes with feedback

1.4 TESTABILITY IN DIGITAL SYSTEMS

When testing a digital logic device, a stimulus is applied to the inputs of the device and its response is checked to establish that it is performing correctly. The input stimulus is referred to as a test pattern.

In general the response of the device at its normal output pins is observed. However, it may be that the device is specially configured during the test, to permit observation of some internal nodes which would generally not be accessible to the user.
The response of the device is evaluated by comparing it to an expected response, which may be generated by measuring the response of a known good device, or simulated in a computer. Even if the device passes the test, it cannot be categorically said that it is a “good” device. The only conclusion that can be drawn from the device passing a test, is that the device does not contain any of the faults for which it was tested. It is important to grasp this point. A device may contain a huge number of potential faults, some of which may even mask each other under specified operating conditions. The designer can only be sure that the device is 100% good if it has been 100% tested, which is rarely possible in real life systems.

1.4.1 Test Vector Generation

As there is no way of accessing most of the logic, the internals of the device cannot be directly probed. Owing to this, a way of generating tests which when applied to the inputs of a circuit, give a set of signals indicating whether the device is good or faulty is needed. The set of stimulus inputs is called a “test vector”. The test vectors distinguish between a good machine and a faulty machine. In combinational logic design flow, the test pattern generation is expensive and is done after the logic design has been completed. Logic synthesis for testability brings the two steps together to better manage the complexity of test generation.

1.4.2 Testing Principle

The principle of testing a digital circuit is shown in Figure 1.14.
**Figure 1.14 Testing method**

When a sequence of inputs is applied to circuits, the output response is observed. The response is compared with a precomputed or “expected” response. Any discrepancy is said to constitute an error.

1.5 **DESIGN FOR TESTABILITY**

Recent advances in LSI/VLSI technology have led to packages of increasing size and complexity. Besides the considerable problem of testing the packages by themselves, the incorporation of these into larger designs has increased the cost of test generation exponentially. In many cases, the cost of test generation and fault simulation, which is needed as a tool to determine how well the tests perform, has reached a practical limit. One of the approaches to solving the problem is to constrain the design in a way that
makes test generation and diagnosis easier. This is widely known as “design for testability”. A testable circuit has the following properties, namely,

- A circuit can be easily put in the desired initial state.
- The internal state of the circuit can be easily controlled by the application of test patterns to the primary inputs of the circuit.
- The internal state of the circuit can be uniquely identified through the primary outputs of the circuit or through the use of special test points.

Testability is achieved by using extra logic and/or test points. The techniques for incorporating testability into circuits fall into two categories, namely, design guidelines to be followed in order to make the circuits testable and systematic procedures or structures aimed at producing testable circuits.

1.5.1 Controllability and Observability

The two key concepts in designing for testability are controllability and observability. Controllability refers to the ability to apply test patterns to the inputs of the sub-circuit via the primary inputs of the circuit. In order to enhance the controllability of the circuit, the states which cannot be controlled directly from its primary inputs have to be reduced. Observability refers to the ability to observe the response of the sub-circuit via the primary outputs of the circuits or at some other points of output. In general, the easiest way to increase controllability/observability is to add some control gates and control terminals (controllability) or to add some output terminals (observability) for testing purposes.
1.5.2 Design of Testable Combinational Logic Circuits

A number of design procedures have been proposed in recent years for the realisation of easily testable combinational logic circuits. The prime objective of these design procedures is to minimize the number of fault detection tests and/or simplify the generation of such tests.

1.6 BACKGROUND OF THE THESIS
1.6.1 Reed-Muller Expansion Technique

This technique can be used to realize any arbitrary n-variable Boolean function using AND and EX-OR gates only. The circuit so designed has the following properties, (Parag Lala 1990).

- If the primary input leads are fault-free, then at most \((n+4)\) tests are required to detect all the single stuck-at faults in the circuit.

- If there are faults at the primary input leads as well, then the number of tests required is \((n+4) + 2n_e\), where \(n_e\) is the number of input variables that appear an even number of times in the product terms of the Reed-Muller expansion. However, by adding an extra AND gate within its output being made observable, the additional \(2n_e\) tests can be removed. The input to the AND gate are those inputs appearing an even number of times in the Reed-Muller product terms.

For a three–variable function of the Reed-Muller expansion is ,

\[
f(W, X, Y) = a_0 \oplus a_1 W \oplus a_2 X \oplus a_3 Y \oplus a_4 WX \oplus a_5 WY \oplus a_6 XY \oplus a_7 WXY
\]
The constant $a_i$ for a three-variable Reed-Muller expansion may be computed using the following rules (Parag Lala 1990):

$$a_0 = f_0$$

$$a_1 = f_0 \oplus f_4$$

$$a_2 = f_0 \oplus f_2$$

$$a_3 = f_0 \oplus f_1$$

$$a_4 = f_0 \oplus f_2 \oplus f_4 \oplus f_6$$

$$a_5 = f_0 \oplus f_1 \oplus f_4 \oplus f_5$$

$$a_6 = f_0 \oplus f_1 \oplus f_2 \oplus f_3$$

$$a_7 = f_0 \oplus f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5 \oplus f_6 \oplus f_7$$

where $f_i$'s are the output values of the minterms obtained from the truth table. This procedure can be easily extended for higher-order expansions.

Let us now consider the Boolean function,

$$f(W,X,Y) = WX + \overline{WY} + \overline{XY}$$

The Reed-Muller expansion of the function is,

$$f(W,X,Y) = 1 \oplus X \oplus WX \oplus WY \oplus XY$$

A direct implementation of the function is shown in Figure 1.15.
Reed-Muller (RM) logic, as an alternative means for logic design, is based on two basic operations in modulo-2 arithmetic, namely, modulo-2 addition and modulo-2 multiplication. These two operations are identical to Boolean Exclusive-OR (EX-OR) operation and AND operation. The resulting algebra is that of the finite or Galois field GF. This mode of representation supports the familiar mathematical operations such as matrices, transforms and polynomials. The resulting circuits are much easier to test than their counterparts in the conventional Boolean domain.

Unlike the situations in the Boolean domain, there exists a great deal of canonical expansions in the RM domain. This makes it more difficult to minimize a logic function in the RM domain than in the Boolean domain. In addition, for a given logic function, its canonical expansions in the Boolean domain corresponds directly to a truth table that completely specifies this function. For a logic function in the RM domain, there is no similar relationship between its canonical expansions and a truth table. This means that, no general means, like a truth table in the Boolean domain, can be used to specify a logic problem in the RM domain.
This technique was first introduced by Zhegalkin in 1927. He described a kind of ESOP form, which included only uncomplemented Boolean variables. Later, Reed and Muller used ESOP for logic circuit design and error detection. From this time, ESOP expressions have been called Reed-Muller expressions (expansions, representations, forms, etc.) in the literature because Zhegalkin's work, written in Russian, was unknown.

A lot of precious work about Reed-Muller logic, especially in theory, has been done in the past. Some results may be feasible for commercial products. The first automatic logic synthesis system to make use of the mixed-polarity Reed-Muller expansion is GATEMAP. GATEMAP concurrently maintains three different representations of each logic function throughout all stages of its operation. These three representations are Sum of the products of the function, Sum of the products of the inverse of the function and Mixed polarity of the function.

The best one is chosen to implement the function. A wide variety of functions are tested by GATEMAP. Reed-Muller equations are found to be normally of comparable size to the Sum-of-Products. The exceptions are where, the Reed-Muller equations are significantly smaller. Another main advantage of Reed-Muller logic is that, when a logic circuit is realized with Reed-Muller expansion in two level fixed polarity form, it requires a short test set to detect a single stuck-at fault (stuck-at 0 and stuck-at 1) in the circuit, and the test set is independent of its actual function being realized.

1.7 OBJECTIVES OF THE RESEARCH

The literature survey shows that research is being carried out for about four decades in the field of fault detection in digital circuits, especially with the network structure of Reed-Muller canonical form. Test vectors for determining various types of faults such as the basic single stuck-at, multiple stuck-at, bridging and stuck-open categories had been tried out with
modifications of the basic ESOP networks. The cardinality of the test vectors proposed by many authors become prohibitively excessive for a large number of input variables.

In the present thesis, an attempt has been made to propose the following for the ESOP Reed-Muller canonical circuits.

1. A compact universal test set with \((n+5)\) vectors. Each of the vectors is \((n+nzI+3)\) long, ‘\(n\)’ being the number of data inputs and ‘\(nzI\)’ the number of complementary inputs.

2. Five easily testable network structures conforming to the test set mentioned above with better fault identification capability.

3. The application of the network structures and test sets of single stuck-at faults for three other types of faults also namely double stuck-at, AND-bridging and OR-bridging faults.

Ten random ESOP functions have been considered from three variables to twelve variables. Four types of faults viz. single stuck-at, double stuck-at, AND-bridging and OR-bridging faults have been simulated using MATLAB. All the proposed structures are of tree type structure to reduce propagation delay.

Two quantification indices, called identifiability factor and distinguishability factor have been computed for the ten random functions to compare the performances of the various structures. A result compaction technique enables the responses of the circuit under test for various input patterns to be represented in decimal form. This scheme requires only a little memory space and is also very convenient to interpret the results.
While the literature survey indicates a separate set of test vectors specifically for the detection of multiple stuck-at faults, an attempt has been made in the present research to explore the possibility of the applicability of the same network structure and test-set of single stuck-at fault detection to double faults, AND-bridging faults and also OR-bridging faults. Only two lines at a time have been assumed to be held at stuck-at-0 or suck-at-1 for the multiple stuck-at and bridging faults. For the double stuck-at faults, all the four possible combinations (0,0),(0,1),(1,0) and (1,1) have been tried out for each pair of faulted lines. The bridging faults have post-fault values same as the logical AND-ing or logical OR-ing of the pre-fault values of the lines considered.

The approach flow diagram is shown in the Figure 1.16.

![Figure 1.16 Approach flow diagram](image-url)
The salient features of the proposed schemes are listed below, namely,

- Function independent test set
- Easy testability analysis of the given ESOP RMC circuit, with clear identification and location of various faults and easy implementation
- Single network structure to detect four types of faults
- Enhancement in the performance of the diagnosis system
- Faster processing because of tree structure

1.8 ORGANISATION OF THE THESIS

Chapter 1 gives an overall introduction to the Exclusive-OR sum of product Reed-Muller Canonical form of Boolean logic functions. Chapter 2 briefs the literature survey on single stuck-at faults, multiple stuck-at faults and bridging faults. Chapter 3 describes the materials and methods of the research. Chapter 4 deals with the reference structure and details the analysis and diagnosis of all the four types of faults viz. single stuck-at, double stuck-at, AND-bridging and OR-bridging faults. Chapters 5 to 9 discuss the five proposed structures and analyze the results. Chapter 10 consolidates the results of all the proposed structures. Chapter 11 gives the scope for future research and conclusion.