Reversible Circuits

Protik Datta

April 2023

1 Introduction

The main purpose of studying reversible circuits is the theoretical possibility of being able to perform computations at almost zero energy cost.
Bennett (https://knowen-production.s3.amazonaws.com/uploads/attachment/file/3087/bennett.pdf) demonstrated that it is possible to construct a reversible turing machine that does not destroy information and thus can be operated with very little energy.
In reversible computation there is an equal number of inputs and outputs.

2 Reversible Circuits

A logical circuit is reversible if it is a bijective (one-to-one) function, which means each input combination is uniquely mapped with an output combination. This effectively enables us to determine the input using the output which is not always possible in classical computation.
Reversible Circuits are built using gates similar to how non-reversible circuits are made. These gates are reversible in nature as well.
A very well-known reversible gate is the Toffoli gate. The Toffoli gate has 2 control lines and 1 target line, if the controls are represented using a, b and the target is represented using c. After passing through the Toffoli gate, the target line becomes $c \oplus a \wedge b$ where $\oplus$ represents the XOR operation and $\wedge$ represents the AND operation. In simpler terms, if a and b are 1, c gets flipped.

3 Faults

3.1 What is a Fault?

A fault is a disturbance in the system which causes the circuit to deviate from it's intended behaviour. A certain input is said to detect a fault when it's resultant output is different in the absence and presence of the fault. Faults may or may not be detectable. Some faults do not affect the final output of the circuit and they are called redundant faults.

3.2 Types of Faults:

There are many types of faults but for our purposes we will be focusing on these three types:

3.2.1 Single Missing Gate Fault

1. **Definition**
   Abbreviated as SMGF, as the name suggests, single missing gate fault refers to the absence of a single gate from the given circuit.

2. **Detection**
   An input which produces 1's at every target line for the gate in question can be used to detect Single Missing Gate Faults for that gate. For a n-Gate circuit, there can be a maximum of n SMGFs.

3.2.2 Partial Missing Gate Faults

1. **Definition**
   Abbreviated as PMGF, partial missing gate fault refers to the absence of one or more controls from a given gate. For our purposes, we will only be talking about 1° partial missing gate faults.

2. **Detection**
   If a given gate has n-controls, an input that produces 1 at n-1 such controls can be used to detect partial missing gate faults for that gate. For a n-Gate circuit and every gate having c_i controls where $i \in [1, n]$, the total number of 1° PMGFs can be written as $\sum_{i=1}^{n} c_i$.

3.2.3 Multiple Missing Gate Faults

1. **Definition**
   Abbreviated as MMGF, multiple missing gate fault refers to the absence of more than one, continuous gates for a circuit.

2. **Detection**
   If a given gate has n-controls, an input that produces 1 at n-1 such controls can be used to detect partial missing gate faults for that gate. For a n-Gate circuit and every gate having c_i controls where $i \in [1, n]$, the total number of 1° PMGFs can be written as $\sum_{i=1}^{n} c_i$.

4 Motivation

The number of input combination grows exponentially with the number of input lines. For example for 5 input lines there can be 2^5 input combinations. As the number of input line grows, it becomes infeasible to detect flaws in the circuit efficiently. Hence there arises a need for a minimal test set of input vectors that covers most if not all faults possible for a given circuit.
Finding such a minimal set is a variation of the set-cover problem. The set-cover problem is a NP-Hard problem which implies there is no optimal way to determine if a such a given input vector is the minimal set or not.
We will only be considering SMGF, 1° PMGF and MMGF. Not all MMGFs are detectable, some can go completely undetected.

5 Approaches

5.1 First Approach

Lets say the circuit has n input lines and g gates. We start off by simulating the circuit and storing the SMGF, PMGF and MMGF faults that can be identified using the input. This pre-computation takes 2^n cycles since there are 2^n possible input vectors.

```
fault_table <- init with an array containing 2 fields, the input vector and the faults ideitified
    by it

fault_table <- sort(fault_table) // based on no of faults identified



selection <- {}
answer <- []

while reverse_sorted_table is not empty:
    most_faults_identified <- fault_table.first["faults"]
    answer = answer + fault_table.first["circuit_input"]

    selection = selection U most_faults_identified

    for index, element in enumerate(fault_table):
        reverse_sorted_table[index]["faults"] <- element["faults"] - most_faults_identified
        if fault_table[index]["faults"]) is empty:
            reverse_sorted_table[index] <- null

    fault_table <- [elem in fault_table such that elem is not null]

    fault_table <- sort(fault_table) // based on no of faults identified
```