Abstract

Fault tolerance has been a design challenge in critical electronic systems however this challenge is now migrating into everyday non-critical systems where engineers must aim to design reliable systems on unreliable components. Existing fault tolerance approaches only provide limited levels of reliability as inherent architectural constraints are placed on the number and type of faults that can be tolerate. Therefore, there is a need to explore new fault tolerant methods and analyse the overhead.

This project provides an exhaustive comparison of different Fault tolerant techniques when they are embedded into memory designs. It begins with a review of the factors which cause the creation of soft errors and how they contribute to faults in the memory designs. Furthermore it examines the detection and correction abilities of various techniques. The detection and correction abilities are compared from an implementation point of view. The delay and area penalties for implementing error correction of these techniques are analysed using various simulation analysis.
In addition, it explores the ability of each fault tolerant method in improving the Yield. It also investigates the use of the correction capabilities of different error correcting codes as a way to correct the defects which are present in the wafer during the manufacture process.

Finally, we introduce some new metrics which are able to combine the fault tolerant abilities in conjunction with the contribution to the Yield so the advantages and the disadvantages of each technique can be analysed. [1]

[1] Furthermore during the project a novel fault tolerant technique was created and submitted to IEEE Transaction Device and materials reliability

Chapter 1

Introduction

Constant advances in manufacturing yield and field reliability are important enabling factors for electronic devices pervading our lives, from medical to consumer electronics, from the railway scenario to the automotive/avionics one. Digital design techniques have exploded in the last decade followed by other technological achievements. Almost all applications in modern society revolve mostly or partially around hardware applications. At the same time, both technology and architectures are today at a turning point; many ideas are being proposed to postpone the end of Moore's law such as extending CMOS technology as well as finding alternatives to it like CNTFET, QCA etc, while at the architectural level, the spin towards higher frequencies has been replaced by the trend of including many cores on a single die.

With the increasing number of transistors, low power and an efficient interconnect are the main requirements of modern hardware design. However as we integrate more and more transistors the designs are vulnerable to soft errors due to scaled technology and cross talk and other effects. A crucial block in any design is memory. Memories are the densest part of the system. The reliability of the system is highly depended on the reliability of the memory. To enhance the reliability particularly the ability of memories against soft errors needs to be addressed. This is the main concept which will be analysed in this project.

Soft errors are the non-permanent errors which exist in digital designs and do not cause any lasting permanent damage. The definition of fault tolerant techniques is the techniques which are created with the purpose of avoiding any problem during the working life of a system. The method which is used to achieve it varies between each technique. They even try to correct any possible error or to detect it and to inform the system of the appearance of this error.

Various techniques have been suggested to increase the reliability of the systems. In this project a detailed analysis of the major techniques which are used against soft errors will be addressed. The independent analysis of each technique has as its purpose to discover the ability of each technique against soft errors. For the analysis 2 metric the MTTF was introduced (Mean time to failure) and the MTTC (Mean time to catastrophe). The metrics and the simulations of fault tolerant design were performed using C models. Multiple simulations were performed to explore useful conclusions according to the reliability provided by each technique.

With all the fault tolerant methods examined, a synopsis tool was used to synthesize VHDL modes of the design. In this way it was able to calculate the overhead for each technique. The overhead was studied in terms of area, power and delay. Another metric was introduced which was the Cost, which is defined as the multiplication of the three parameters of the overhead. The MTTF and MMTC was express in terms of the Cost, so these metrics became more realistic.

Another significant study was made in the area of the yield and the contribution which will exist if the fault tolerant techniques are used to improve the yield. The yield is called the number of acceptable chips divided by the chips which are produced from the wafer during the manufacturing process. It will explore the ability of these methods to improve the yield with all the advantages and disadvantages which exist via exhaustive use of simulation using C language. The simulation examined the number of good chips for different numbers of defects at wafer using various scenarios. Furthermore it tested the improvement in yield when it was used with additional rows and columns (in conjunction with fault tolerant techniques).

Finally during the project a novel fault tolerant technique was created. This technique was called in this project RMC Extended and it was based on an existing technique which was also examined in the project.

1.1 Aims and Oblectives

The project's aim was to create an exhaustive comparison between the different fault tolerant techniques according to their MTTC and MTTF per Cost values. For a successful comparison it constructed a detailed fault tolerant model in C language and also the VHDL model (to calculate the Cost values).Furthermore it examined the usage of the fault tolerant techniques to improve the Yield.

1: To research all the existing techniques according to their capability to correct/detect errors and the overhead of each technique.

2: To examine the new technique and examine its capability of error correction and detection and also explore its advantages and disadvantages which followed the use of this method.

3: To design using VHDL language the fault tolerant technique and synthesise using Synopsis tool. Estimate area,power,delay using the same tool. Using the measured performance parameters(area,power,delay) calculate the Cost for each technique.

4: To create a C language model for each of the techniques and exhaustively simulate the metrics MTTC per Cost and the MTTF per Cost. From the results of these simulations it was able to compare the fault tolerant technique abilities.

5: To produce a C language model to simulate the improvement of the Yield by using each fault tolerant method. For different numbers of error per wafer it would take measures according to the good chips. It will also compare the numbers of good chips with reliability and

the chips which become good chips through the use of the fault tolerant abilities.

6: To create a C language model to simulate the improvement of the Yield by using additional rows and additional columns in conjunction with fault tolerant methods. In addition to the preview simulation, it will use additional rows and columns to investigate the contribution to the yield.

7: To use a metrics benefit and benefit base technique to find comparisons with different fault tolerant methods.

8: To create a comparison between the proposed technique (RMC Extended Reduced) with existing methods on which it is based.

1.2 Outline of the project

This thesis will first introduce some background knowledge and previous work done by the others in the area of fault tolerance methods and coding theory.
A detail description of soft error and it's effects in a memory design are also described. Also in Chapter 2 presents different fault tolerant techniques.
Chapter 3 describes the implementation of models and design flow.Also the algorithm used in this thesis are explained.Furthermore a new Coding methodology is introduced in the same chapter.
Moreover experimental results is given in Chapter 4.The results are analysed by defining new metrics. Conclusion and critical analysis of the thesis are given in Chapter 5.
Finally Chapter 6 presents possible future work.

Chapter 2

Background Survey

The background survey is divided into two parts. In the first part we will try to give an explanation of the soft errors, how they are created and what are the effects at digital design circuits and more particularly at memory designs. In the second part in Sections 2.5 and 2.6 some basic parameters are described regarding reliability and Yield. The remaining sections describe different fault tolerant techniques, explain how they work and present their capabilities against errors.

2.1 Soft Errors

The main characteristics of modern designs are the decrease in their size in conjunction with low power requirements and high density functionality. For the satisfaction of the above requirements, the designs are vulnerable to soft errors. Soft errors or Single Event Upsets (SEU) are used by the author to describe the non-permanent errors which are present in digital designs. SEU does not cause any long-term damage to the circuit [4]. When a high energy radiation event appears it is possible for multiple bits upset (MBU) to be created. In Section 2.2 the behaviour of the radiation at silicon will be described and in Section 2.4 we will describe the behaviour at the SRAM.

2.2 Behaviour of radiation at silicon devices

An event upset occurs in an electronic circuit when a semiconductor is charged by ionizing radiation. Ionizing radiation can be caused either by direct ionizing or by indirect ionizing. The Linear Energy Transfer (LET) of an ion defines the magnitude of the disturbance to this ion. The mass, the energy of the particle and the material which the ion strikes, define the LET. In general, charge collection arises when mainly neutron reaction events and fewer alpha particle events exist.
Figure 2.1 illustrates an example of a reverse bias junction in the case of an ionizing radiation event and the effects which occur. The reason for its use in the example, is that a reverse bias junction is the most vulnerable part of the semiconductor device. When ionizing radiation strikes the material, an electron hole pair with high carrier concentration is activated. When the ionization approaches the depletion region of the device, electric fields collect carriers

extremely quickly. As a result, a voltage and current transient and funnel shape are created at this point as shown in Figure 2.1(a). Furthermore, the depletion region is increased to the substrate because of the increase of drift collection by the funnel. The amplified distortion of the funnel means a drop in the substrate doping; this process step is called the "prompt" collection phase. At the end of this procedure, the domination of the collection process is ignited by diffusion. This step is shown in Figure 2.1(b). In the last step the depletion region collects the electrons via diffusion. This process is continuous until all the carriers are collected. The last step is shown in Figure 2.1(c). Finally the current for the three states at biased-junction is shown in Figure 2.1(d) [4].

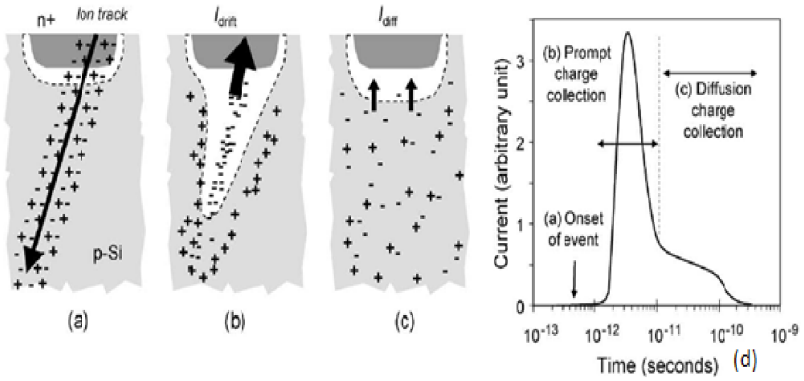


Figure 2.1: Reversed Biased-Junction [4]

2.3 Radiation sources

The main sources for radiation which affect soft errors are the Alpha particles and hight energy cosmic rays. Specifically, cosmic rays are divided into high energy cosmic rays and low energy cosmic rays.

- As far as alpha particles are concerned, they are caused through nuclear decay of unstable isotopes. The above particles are comprised of two neutrons and two protons. The primary isotopes that are responsible for alpha particles are Uranium and thorium. This procedure creates the Helium ionized atom. The range of energy emitted by the alpha particles is in the range 4-9 Mev. During the strike of an alpha particle on the material, a loss in its kinetic energy is observed mainly because it interacts with the material electrons, leaving behind a trail of ionization. The alpha particle passes through the material, attenuates it and finally the material "stops" the particle. The distance travelled to the material is less than 100 μm with an energy of 10 MeV. The protection of the semiconductor device can be achieved through shielding or by using high impurity materials. These methods are able to reduce the number of alpha particles [4].

- An additional important source of soft errors are high energy neutrons which are caused by cosmic rays. Primary cosmic rays react with the atmosphere of the Earth and produce complex cascades of secondary particles due to the strong interaction. They penetrate the atmosphere, generating tertiary particle cascades. From the primary flux only a few

of them (less that 1%) reach sea level. The major particles at the terrestrial altitude consist of protons, neutrons, muons and pions. Neutrons are stable due to the fact that coulombic contact with the atmosphere reduces the effects of electrons and pions and muons have a shorter life span with a high flux. The reactions of the neutrons cause higher LETs to semiconductor devices than the alpha particles. As a result, neutrons are the most familiar type of cosmic radiation which create single event upsets or multiple event upsets. So the most important factors for the creation of soft errors are the cosmic events and the neutron reactions caused by them. Unfortunately the reduction of the cosmic neutrons from cosmic rays is not possible with any method at the IC device [4],[3].

- Finally the last source of radiation is the low energy cosmic ray. Low energy neutrons from cosmic rays, in conjunction with boron are induced from secondary radiation which ionizes particles in semiconductor devices. Despite the third source being produced by cosmic rays (as in the second source), it is referent to low energy neutrons by less than 1Mev. Boron is used to a great extent in the process of the manufacture of semiconductor devices such as the p-type dopant. Furthermore Boron is made up of two isotopes: ^{11}B and ^{10}B. The latter becomes unstable in the presence of a neutron. As shown in Figure 2.2 below when a neutron strikes an isotope, ^{10}B nucleus breaks and creates ^{7}Li (and a gamma photon from ^{7}Li recoil) and an alpha particle [4],[3].

Both the produced alpha particle and the ^{7}Li recoil are able to cause soft errors (as a result of the high peak LET). In modern electronics, where low power is a requirement, the ^{7}Li recoil and the alpha particle increase the ability to cause soft errors.

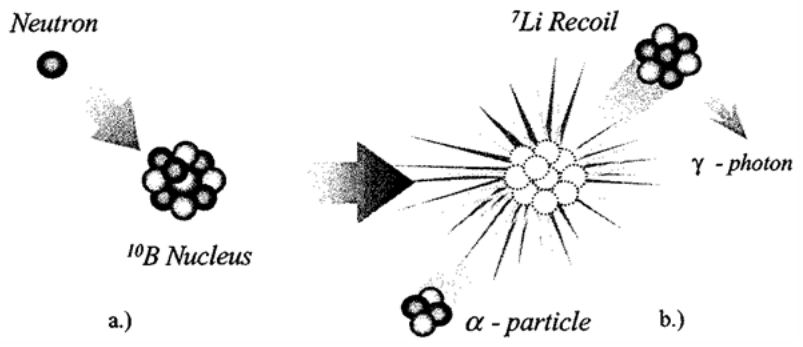


Figure 2.2: Fission of ^{10}B [3]

2.4 Static Random Access Memory (SRAM)

Static random access memory (SRAM) is a standard memory which is situated in each computing system and is extremely sensitive. In the case where an energetic particle strikes a reverse biased junction of the transistor, the specific transistor becomes turn on/off.The charge collected due to the junction results in a transient current at the transistor. In the flow of the specific current to the transistor, the p transistor (which is shown in Figure 2.3) tries to stabilise the current which is induced by the particle with the outcome of the source current. The capability of the p transistor to supply current is limited and induces a voltage drop at its drain. The drop in voltage as a consequence of the current transient is the mechanism, which produces soft errors in the SRAM cell. The sensitive area in each SRAM cell is the 4 transistor area. An essential factor for charge collection is the position of the junction and it is more specific if it is located inside-the-well or outside-the-well. The most sensitive area is outside-the-well. The reason is the capability of the diffusion back to the drain junction as a product of the deep charge deposited in the substrate. The charge deposited deep in the substrate can diffuse back to the drain junction. On the other hand when an off strike is present inside-the-well the primary drift current pulls the struck node potential downwards, starting the upset procedure. As the transient continues, holes placed in the p-well are collected at the p-well ties, raising the well potential and causing the injection of electrons via the source. For small geometries, the inside-the-well "off" strike can develop into a significant mechanism [8].

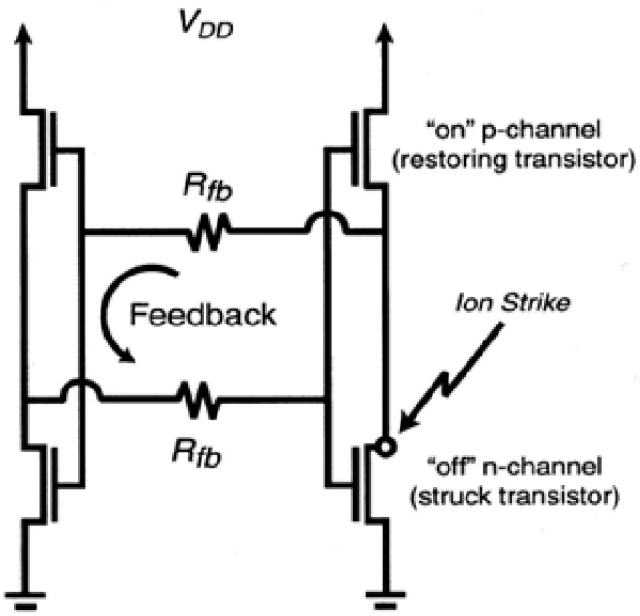


Figure 2.3: Basic implementation of SRAM memory [8]

2.5 Reliability-Failure Rate-Mean time to Fail

The reliability (R) of a modern digital system is a crucial factor for a system. The reliability of a system is defined as the probability of a system to experience no failures in a defined time.
The high complexity in the architecture hardware, in conjunction with the high scaling of CMOS technology creates a high number of hardware failures in a modern digital system. This is the reason for the high requirement of reliability in the system.

The hardware failures are follow the bathtub curve model as is shown in the Figure 2.4.From the Figure, it is obvious that in near life of hardware's system has the maximum rate of failures and is follow by a period which is the useful life of the system and has lower rate of failures. At this period the faults remain stable to a lower rate. After the useful life of the system, at the last period failures are increase according the system become older [19],[13].

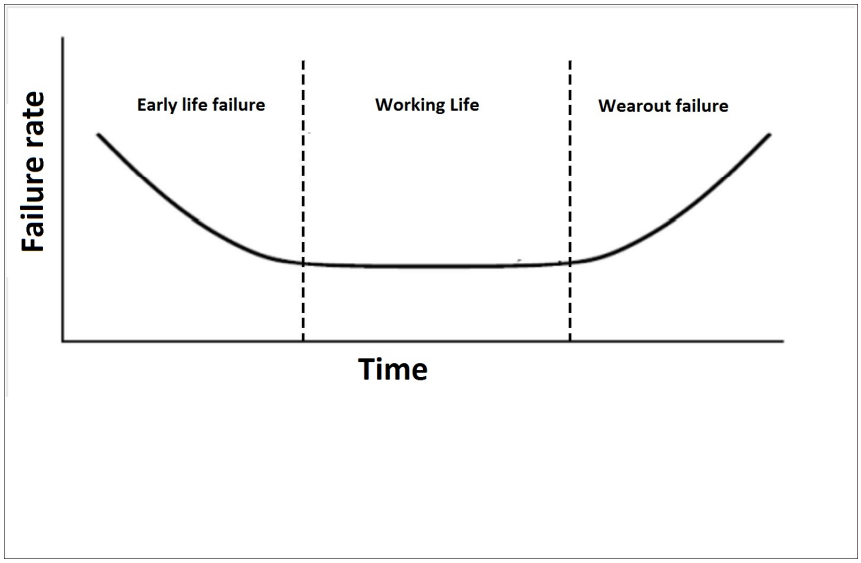


Figure 2.4: bathtub curve model

The failure rate (λ) which referred above is defined as the number of components in which failures in a system divided by the number of the components which are not suffer by any failure.

$$Z(t) = \frac{Number\ of\ failing\ components\ per\ unit\ time\ at\ time\ t}{number\ of\ surviving\ components\ at\ time\ t} \quad (2.1)$$

Furthermore the reliability in terms of failure rate can express as follow

$$\int_0^t Z(t)dt = \int_0^t \frac{f(t)}{R(t)}dt = -\int_{R(t)}^{R(t)} \frac{dR(t)}{R(t)} = -ln\frac{R(t)}{R(0)} \tag{2.2}$$

Where the f(t) is the density of the failures probability density functions and R(t) is the reliability in the time [19]. Another important factor of the reliability is the Mean time to failure (MTTF). Is express the average time for a system (or a component of the system) to caused a fail and the mathematical expression is given by the equation 2.3.

$$MTTF = \int_0^\infty t.\lambda e^{-\lambda.t}dt = \int_0^\infty e^{-\lambda.t}dt = \frac{1}{\lambda} \tag{2.3}$$

2.6 Yield

During the manufacture process of chips there are a lot of parameters like impurities of the material in the wafer or dust particles which cause defects at the wafer. The word defect describes the physical imperfection in the processed wafer. During the project the defects are called errors. Moreover the usual defects which encountered at the chips are: the missing contacts, missing conductors, broken conductors, wrong doping level etc. The yield (Y) called the number of acceptable chips divided by the chips which are produced from the wafer during the manufacture process. An example of defect and the calculation of the yield is described below. The Figure 2.5 illustrates a wafer with defects (the defects in the wafer are shown using a dot)[5].

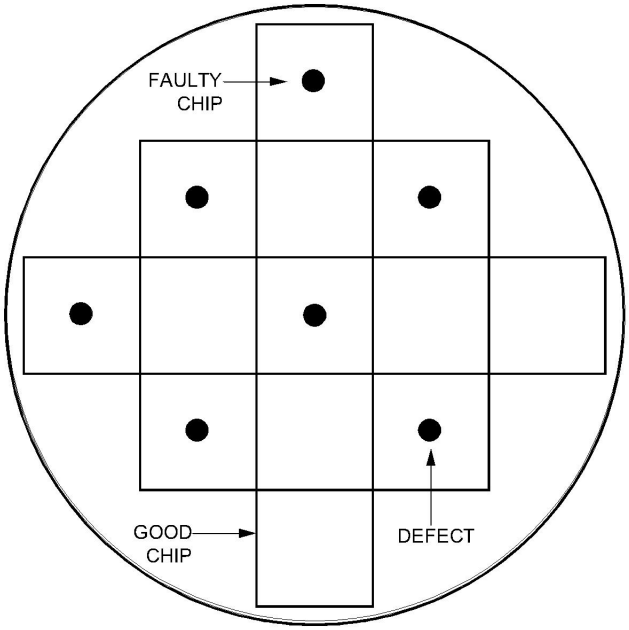


Figure 2.5: Wafer with Defects

To provide a better understanding a simple example of Yield has been presented. In the wafer in Figure 5 there are 7 defects (consider the wafer as being able to manufacture 13 chips) and there are 6 chips which are free of defects (good chips) so the yield is calculated as 7/13=0.55.

2.7 Techniques for Soft error Detection /Correction

A lot of techniques have been created which try to detect or event to correct, the presence of a soft error in the design. The most significant techniques are described in this chapter.

2.7.1 Duplicated With Comparison (DWC)

An important technique for Single Error Detection (SED) is Duplication with Comparison. In the above mentioned technique the design is copied into two modules (Module A and Module B). It is obvious that both of the modules are expected to have an identical output value (because it is the same design).If an error occurs in one of the modules its output would be different. So in this case the two modules will have different outputs and the mismatch of the two values will be detected by the comparator. Unfortunately because there is only one module with errors and one with no errors, the design is not able to decide which one is the right module. As a result, to achieve the correct output a recalculation of the input value is needed. Despite the fact that with the above method a high defect coverage is achieved, there are circumstances where an error is not able to be detected. For example in the low probability case in which both modules suffer from an error at the same time, the error is not able to be detected. Also another disadvantage of the DWC is the very high overhead of 100%.

However, the DWC is ideal when there is no time constraint and the fault rate is high. The DWC can considerably increase the time domain when used with roll-forward and rollback techniques. (The explanation of the rollback techniques is where the system rolls back in the presence of a failure. The roll-forward technique is the method whereby the system roll-forward to an error free point under specific scenarios when an error is detected. More particularly, when the DWC technique is used in conjunction with the roll-forward technique, a better performance is achieved than rollback without any important hardware overhead [16].

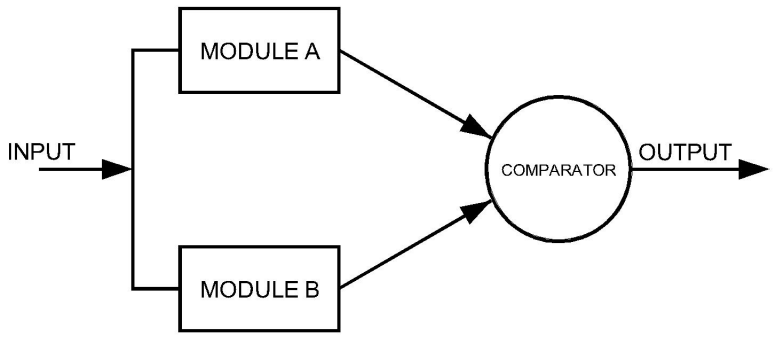


Figure 2.6: Basic diagram of a DWC method

2.7.2 Triple Modular Redundancy (TMR)

In the TMR reliability model the design is copied into three identical modules. Each time an input enters the design the three identical modules are activated. It is assumed that the voter is ideal(no possibility of presenting any error by the voter). If there is any mismatch in the modules this means that an error has occurred in the design. The voter chooses the majority of the module's output as the fault free output of the design.

The TMR is a special case of the N-modular hardware redundancy where the module is copied N times. When N modular redundancy is used with majority voting, it offers the ability to correct faults in C different systems if $N \geqslant 2C + 1$ [9].
This technique is capable of correcting any single design error. In the case of an error occurring, it can detect and determine immediately the fault free output. The main disadvantage of the TMR is the huge overhead of 200%.

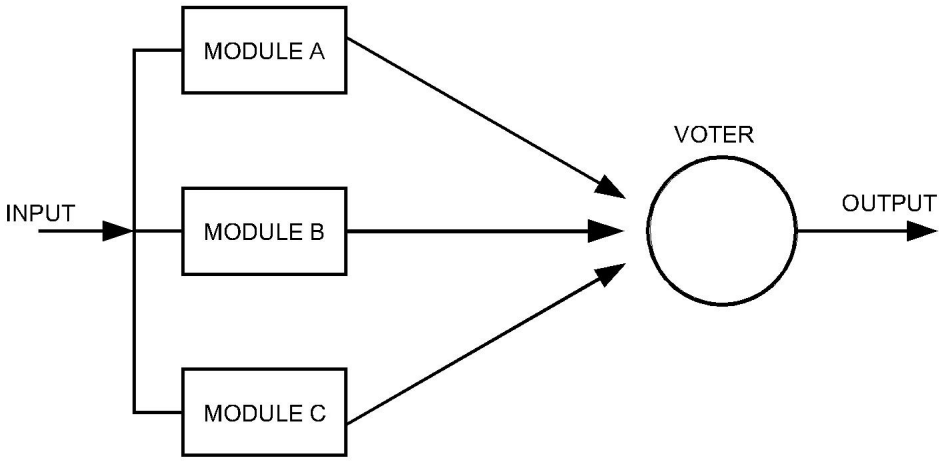


Figure 2.7: Basic diagram of a TMR method

2.7.3 Parity check code

The most common error detection method is the parity check error detection method. The above technique is comprised of the odd and the even parity check error detection method. With this technique an extra bit (parity bit) is added to each word data.

For an even parity check error detection, the parity bit is calculated by exclusive OR the bits of the data word. For example for a 6 bits data word of 110011 the calculation of a bit word is as follows 2.4:

$$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \tag{2.4}$$

Consequently the parity bit in this case is equal to zero. On the other hand when an odd Parity check error detection is used the calculation of the parity bit is calculated by using exclusives OR and exclusive NOR gate as shown below :

$$1 \oplus 1 \oplus 0 \bar{\oplus} 0 \oplus 1 \oplus 1 \tag{2.5}$$

The parity bit for odd parity method is equal to 1.When the parity bit is calculated it is added to the word's packet as a head as shown in the Figure 2.8.

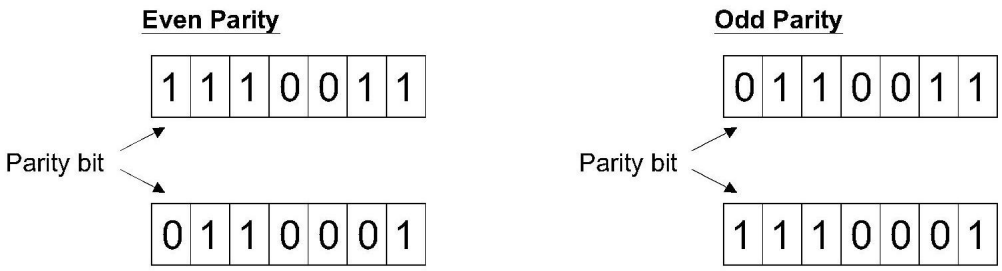


Figure 2.8: Parity check method

The parity bit for the odd parity method is equal to 1.When the parity bit is calculated it is added to the word's packet as a head as shown in the Figure 2.8.

After the calculation of the parity bit the data is sent to the receiver. When the receiver takes the data a similar procedure is executed. The receiver recalculates the parity bit according to the data word and compares the received parity bit with the calculated parity bit. If the received bit is not the same as the bit calculated previously, it means that there is an error in the data word otherwise the data are error free [12].

The main advantage of the parity bit method is the low overhead requirement (only one bit per 1 word data). Also despite the fact that the above technique can efficiently detect a single error it is not able to detect more errors. (It can detect odd number of errors but not an even number of errors).

2.7.4 Hamming Code

One of the most widely used coding algorithms is the Hamming code. It is used either for Single Error Correction (SEC) or for Double Error Detection (DED). Except for the data (code word) it also uses parity bits. The number of parity bits which are necessary for encoding the data are defined according to the Hamming rule and expressed as follows:

$$d + p + 1 \leq 2^p \tag{2.6}$$

In this expression d is the number of data bits and p the number of parity bits. When the parity bits append to the data bits, the Hamming code word is created. The Hamming code word is represented by the expression (c,d) where c is equal to the addition of the number of data bits and the number of parity bits.

To implement the Hamming code algorithm, the first step is to find the number of parity bits which are necessary in accordance with the equation 2.6. The next step is to create the generator matrix (G) where G=[I_kP]. The generator matrix is composed of the identity matrix (I) and the parity generator matrix. The generator matrix (G) is presented as follows:

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & P_{10} & ... & P_{1,p-1} \\ 0 & 1 & 0 & 0 & ... & ... & ... \\ 0 & 0 & 1 & 0 & ... & ... & ... \\ 0 & 0 & 0 & 1 & P_{k0} & ... & P_{kn} \end{array}\right]$$

Furthermore, finding the Hamming code word is needed. It is calculated by multiplying the data bits and the generator matrix code using modulo-2 arithmetic as shown in equation 2.7 [11]

$$A.G = [I : A] \tag{2.7}$$

Each line which is produced by the multiplication matrix contains vectors of the form (d1,...dn,p1,...pm) where n is the maximum number of data bits and m the maximum number of parity bits. The real parity bits are generated by the G matrix. The unique selection of column A has the end result that each parity bit represents parity calculation for each subset of d [11].When a receiver is needed to check the received code word (r) it is multiplied by the parity check matrix (H) to find the parity check vector of the syndrome (S) [11],[1].

$$H = [A^T|I] \tag{2.8}$$

$$S = HXr \tag{2.9}$$

Where the syndrome matrix is equal to zero, the code word does not contain any errors. In the opposite case, the code word includes errors. For a single error (only one bit is not zero) at the syndrome matrix, further analysis is needed to find the bit which failed. The writer used the following example to clarify the concept further. The basic Hamming code (7,4) is chosen.

The first step to implement the Hamming code technique is to calculate the number of parity bits which are needed for 7 bits data. Parity bits which are needed for 7 bits data are calculated using equation 2.4, which is equal to 4 bits. The generation of matrix A (from the able parity bits) is shown below

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 2.9: A matrix - Example

The generation of H matrix using equation 2.8 is calculated as shown below

$$H = \left[\begin{array}{cccc|ccc} D_0 & D_1 & D_2 & D_3 & SP_0 & SP_1 & SP_2 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

Figure 2.10: H matrix - Example

In the Figure 2.10 ,$D_0 ... D_3$ are the data bits and the $SP_0 ... SP_2$ are the syndrome parity bits.

The equations of syndrome parities are shown in the following equation:

$$SP_0 = D1 \bigoplus D2 \bigoplus D3 \tag{2.10}$$

$$SP_1 = D0 \bigoplus D2 \bigoplus D3 \tag{2.11}$$

$$SP_2 = D0 \bigoplus D1 \bigoplus D3 \tag{2.12}$$

Furthermore, if the H matrix is modified (adding an extra row and extra column) as shown below, error correction is possible. If the syndrome parity bit 3 is equal to 1, it means that the code word includes SEC. If the syndrome parity bit 1 is equal to 0, it means that either there is a double error or there is no error. In addition, if the syndrome parity bit 3 is equal to 0 and one of the other syndrome parity bits is equal to 1 there is a double error.

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & | & 1 & 0 & SP_3 \\ 1 & 0 & 1 & 1 & | & 0 & 1 & SP_3 \\ 1 & 1 & 0 & 1 & | & 0 & 0 & SP_3 \\ - & - & - & - & - & - & - & - \\ 1 & 1 & 0 & 1 & | & 0 & 0 & SP_3 \end{bmatrix}$$

Figure 2.11: H matrix with correction abilities - example

The Hamming code has a moderate overhead which varies (depending on size of the data). For 32 bits data, 6 parity bits are needed and for 16 bits data, 5 parity bits are needed. Hence the overhead of the implementation of the Hamming code is 6 bits for 32 bits data and 5 bits for 16 bits data. The major advantage of the Hamming code is its ability for double error detection or single error correction. The Hamming decoding algorithm is present in the algorithm 2.1. At this project the Hamming code which is used is the Hamming code for 32 bits.

Algorithm 2.1 Hamming Code technique

1: Calculate the transport of A^T
2: Create H matrix (**H=**$[A^T$—**I]**)
3: Calculate Syndrome Parity bit (**S=HXr**)
4: Check if Syndrome bits are equal with 1
5: **if** $SP_3 == 1$ **then**
6: $\Rightarrow$ SED
7: **else if** $SP_3 == 0$ **then**
8: $\Rightarrow$ DED or No error
9: **else if** $SP_0 == 1$ OR $SP_1 == 1$ OR $SP_2 == 1$ AND $SP_3==0$ **then**
10: $\Rightarrow$ DE
11: **end if**

2.7.5 Reed-Muller Codes (RMC) - Reed-Muller Reduced Codes (RMC Reduced)

The Reed-Muller codes are among the oldest data linear binary codes. The form of the RMC is presented as RMC (r,m).

2.7.6 Encoding Reed-Muller

To encode a data word the creation of a matrix generator is needed. The dimensions of the matrix generator Reed-Muller (G_{RMC}) is equal to $K \times n$ for a given form (r,m) where k (y axis dimension) is defined according to equation 2.13.

$$k = \sum_{i=0}^{r} \binom{m}{r} \tag{2.13}$$

X-axis dimension is presented as:

$$n = 2^m \tag{2.14}$$

According to the r value the form of RMC is called r^{th} order of RMC. When it is required to encode a data word the first step is to choose the desired error correction requirements. This is calculated as the Hamming distance between any two rows in the r^{th} order of RMC.

During the formation of the RMC matrix, the matrix is needed to be completed using precise digital values. This is done in order to complete all the necessary orders of the RMC code starting with the order 1 and moving one by one until the r^{th} order is reached. The first order of RMC is a vector of all one values with size 2^m. The next action is to move to the next order of the RMC. For the second order the number of vectors which needs to be constructed is equal to the value of $\binom{m}{2}$.

The second vector is assembled when the first half length (2^{m-1}) is filled with 1 and the remaining with 0. The third order again divides each sequence of 1 or 0 and fills each 2^{m-2} length with 1 and 0 respectively. It is necessary to continue with this technique until a vector's values are equal to only one value of 1 which is immediately followed by one value of 0 as shown in Figure 2.12.

$$G_{RMC} = \begin{bmatrix} \psi(1) = \underbrace{11...1}_{2^m} \\ \psi(x_0) = \underbrace{11...1}_{2^{m-1}}\underbrace{00...0}_{2^{m-1}} \\ \psi(x_1) = \underbrace{11...11}_{2^{m-2}}\underbrace{00...00}_{2^{m-2}} \\ \psi(x_2) = \underbrace{1...1}_{2^{m-3}}\underbrace{0...0}_{2^{m-3}} \\ \psi(x_n) = \underbrace{10....10}_{2^m} \end{bmatrix}$$

Figure 2.12: Generator matrix - Example

When it is needed to create more vectors in order that the matrix is filled, the remaining vectors should be constructed by the multiplication of the previous vectors as shown in Figure 2.13.

$$G_{RMC} = \begin{bmatrix} \psi(0) = 11111111 \\ \psi(X_0) = 11110000 \\ \psi(X_1) = 11001100 \\ \psi(X_2) = 10101010 \\ \psi(X_0X_2) = 11000000 \\ \psi(X_1X_2) = 10001000 \end{bmatrix}$$

Figure 2.13: Generator matrix - Example

To enhance an understanding of the RMC (2,3) code the author has provided the following example. First of all, from equations 2.13 and 2.14 We are able to find the size of the matrix. The first row $\psi(0)$ is all "1". The other rows are shown below and calculated accordingly. It is important to mention that vectors $\psi(X0.X2)$ and $\psi(X1.X2)$ are calculated by the multiplication (one by one of the position) of the vectors X_0,X_2 and X_1,X_2 respectively.

$$G_{RMC(2,3)} = \begin{bmatrix} \psi(0) = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \psi(X_0) = & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \psi(X_1) = & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ \psi(X_2) = & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \psi(X_0X_1) = & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \psi(X_0X_2) = & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \psi(X_1X_2) = & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.14: Generator Matrix of RMC(2,3)

Figure 2.15 shows the various vector definitions such as zero order, first order and second order.

$$\left.\begin{array}{l} \psi(0) = 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\ \psi(X_0) = 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \psi(X_1) = 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \\ \psi(X_2) = 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \\ \psi(X_0X_1) = 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ \psi(X_0X_2) = 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ \psi(X_1X_2) = 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \end{array}\right\} \begin{array}{l} Zero\ order \\ \\ \\ First\ order \\ \\ \\ Second\ order \end{array}$$

Figure 2.15: Examples of the 3 orders of RMC(3,6)

Given (G_{RMC}) and data (m) where m is the binary message, m is multiplied by the matrix

generator. $m * G_{RMC}$ represents the multiplication of the matrix generator .The result of this is the codeword encoding data [6],[18],[17].

2.7.7 Decoding Reed-Muller

The first action when decoding the RMC data is based on the majority voting. When a code word is received (m), it is multiplied by the Generator matrix. The correct bit for each position is the majority of the results of each dot product calculation which are produced. In this way the m_2 vector is created. When the monomial of r degree is calculated, it is moved to the multiplication between the m_2 vector and to $\binom{m}{r}$ matrix. The end result of this multiplication is the vector S. Finally the S vector which has been calculated is added to the received code word. The decoding algorithm of Reed-Muller is shown below [18],[15].The algorithm for RMC is shown in Figure 2.2.

To make the RMC decoding clearer, the author has provided the example below. The RMC(2,3) is described as follows [6].

Starting of the bottom of the matrix
at $\psi(X_1.X_2)$ is missing the $\psi(X_0)$

characteristic polynomial is $\psi(X_0)$ and $\psi(\bar{X}_0)$

$$\left.\begin{array}{l} u.\psi(X_0) = 0 \\ u.\psi(\bar{X}_0) = 0 \end{array}\right\} \rightarrow m = ------0$$

at $\psi(X_0.X_2)$ is missing the $\psi(X_1)$

characteristic polynomial is $\psi(X_1)$ and $\psi(\bar{X}_1)$

$$\left.\begin{array}{l} u.\psi(X_1) = 1 \\ u.\psi(\bar{X}_1) = 1 \end{array}\right\} \rightarrow m = -----10$$

at $\psi(X_0.X_1)$ is missing the $\psi(X_2)$
characteristic polynomial is $\psi(X_2)$ and $\psi(\bar{X}_2)$

$$\left.\begin{array}{l} u.\psi(X_2) = 1 \\ u.\psi(\bar{X}_2) = 1 \end{array}\right\} \rightarrow m = ----110$$

When the first order is finished it recalculates the u as follow

$$\text{u}= \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Once the second order is decoded the process of decoding is moved to the lower order.

Algorithm 2.2 RMC decoding technique

1: "Examine the rows with monomials of degree r.
2: calculate the 2^{m-r} characteristic vectors for the row.
3: Take the dot product of each of these vectors with received message and then XOR the result.
4: **if** the majority of the dot products is 1 **then**
5: set the position in the original message vector corresponding to this row to 1
6: **else**
7: the position in the original message vector corresponding to this row to 0.
8: **end if**
9: When finished with all monomials of degree r we take the vector of length $\binom{m}{r}$ and multiply by the $\binom{m}{r}$ row used to calculate the vector.
10: Add the result to the received message.
11: Proceed to recursively on the rows corresponding to monomials of degree r-1"[1].

The main advantage of the Reed-Muller algorithm is its ability for multiple error correction. During the process of decoding data, it can produce the correct codeword in the presence of $\left\lfloor \frac{2^{m-r}-1}{2} \right\rfloor$ errors.

The RMC method is very powerful for correcting large number of errors however the main disadvantage of the above method is the high overhead. Another important aspect worth mentioning is that in this project when an author uses the RMC method it refers to the RMC (3,6).

2.7.8 Reed-Muller Reduced Codes (RMC Reduced)

Despite the fact that the RMC code is a powerful technique (according to its ability to correct 3 errors at the same time) it suffers from high overheads (in area, power and delay). The high overhead from the use of this technique is due to the need for the construction of a high dimension matrix. Each time a message arrives at the system (which uses the RMC fault tolerant technique) it needs to multiply the message with the matrix. Also the matrix plays an important role in the decoding process of the message as shown from previous examples. It is obvious that despite the high penalties matrix, it is an essential part of this technique.

Many researchers have proposed different ways of reducing the large overhead of the code. The main improvement of the RMC code is to modify the size of the matrix. The improvements, it is argued is that it is able to reduce the size of the matrix without losing any data. As a result the code RMC (3,6) uses, instead of a matrix size of 42X64, is the matrix 42X54 [7],[10].The RMC code which uses the specific implementation in the project is called the RMC Reduced and this dimension of the matrix will be examined in the project.

Chapter 3

Implementation

This chapter starts with the basic Reed-Muller theory and explains the new version.It explains how the new method works and its abilities/achievements. The following sections describe the algorithms used and explain the way which we have calculated the overhead, the MTTF, the MTTC and the Yield.

3.1 Novel Technique - RMC Extended

A new method called RMC Extended which has been created is presented in this section.The main difference between our technique and the existing RMC appears at decoding part. It is important to note that the two techniques are using the same encoding system. We better explain our technique using the RMC(3,6) example.As mentioned in the section about the RMC method, the first step is to decode a message via the Reed-Muller method beginning by multiplying the message with the highest order of the generator matrix and then moving on to the lower order of the Reed-Muller generator.

In the case of RMC(3,6) the size of the matrix which needs to be construct is 42X64 and the 3^{th} order consists of 20 rows with each row having 64 bits. The third order of the RMC(3,6) is presented in Figure 2.14.

$$G_{RMC(3,6)} = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_1.X_2 \\ X_1.X_3 \\ X_1.X_4 \\ X_1.X_5 \\ X_1.X_6 \\ X_2.X_3 \\ X_2.X_4 \\ X_2.X_5 \\ X_2.X_6 \\ X_3.X_4 \\ X_3.X_5 \\ X_3.X_6 \\ X_4.X_5 \\ X_4.X_6 \\ X_5.X_6 \\ X_4.X_6 \\ X_1.X_2.X_3 \\ X_1.X_2.X_4 \\ X_1.X_2.X_5 \\ X_1.X_2.X_6 \\ X_1.X_3.X_4 \\ X_1.X_3.X_5 \\ X_1.X_3.X_6 \\ X_1.X_4.X_5 \\ X_1.X_4.X_6 \\ X_1.X_5.X_6 \\ X_2.X_3.X_4 \\ X_2.X_3.X_5 \\ X_2.X_3.X_6 \\ X_2.X_4.X_5 \\ X_2.X_4.X_6 \\ X_2.X_5.X_6 \\ X_3.X_4.X_5 \\ X_3.X_4.X_6 \\ X_3.X_5.X_6 \\ X_4.X_5.X_6 \end{bmatrix}$$

Figure 3.1: Third order of RMC(3,6)

According the decoding example at the section 3.6.2 starting at the bottom of the matrix at the row $\psi(X4.X5.X6)$ is missing the $\psi(X3)$,$\psi(X2)$,$\psi(X1)$.The characteristic polynomials are:

$\bar{X_1}.\bar{X_2}.\bar{X_3}$
$\bar{X_1}.\bar{X_2}.X_3$
$\bar{X_1}.X_2.\bar{X_3}$
$\bar{X_1}.X_2.X_3$
$X_1.\bar{X_2}.\bar{X_3}$
$X_1.\bar{X_2}.X_3$
$X_1.X_2.\bar{X_3}$
$X_1.X_2.X_3$

Each one bit of the message needs to be calculated.For the first bit the following equation needs to be used:

$$\left.\begin{array}{l} u.(\psi(\bar{X_1}) * \psi(\bar{X_2}) * \psi(\bar{X_3})) \\ u.(\psi(\bar{X_1}) * \psi(\bar{X_2}) * \psi(X_3)) \\ u.(\psi(\bar{X_1}) * \psi(X_2) * \psi(\bar{X_3})) \\ u.(\psi(\bar{X_1}) * \psi(X_2) * \psi(X_3)) \\ u.(\psi(X_1) * \psi(\bar{X_2}) * \psi(\bar{X_3})) \\ u.(\psi(X_1) * \psi(\bar{X_2}) * \psi(X_3)) \\ u.(\psi(X_1) * \psi(X_2) * \psi(\bar{X_3})) \\ u.(\psi(X_1) * \psi(X_2) * \psi(X_3)) \end{array}\right\} m = (1bit)$$

Using the process of decoding the calculation of the free error bit is exact using the majority property. Where 3 errors exist, there will be 5 error free values and 3 error values. By using the majority it will select the value presented to the 5 values (instead of the 3 values).

In the case of the present 4 errors the output of the majority mechanism is unable to give the right result. The reason is that the result of the majority circuit is uncertain because it will need to find the majority but the number of error values and error free values are the same. As a result there is no majority value to produce.

The motivation at this point was to use another hardware mechanism (except the majority mechanism) called tie logic 3.2. With the standard hardware where the number of errors is 4 and the tie system active, the 4^{th} error can be detected.

The two circuits (Majority and Tie circuit) were constructed using the synopsis tool using 180nm technology and are presented in Figure 3.3 and 3.4.

The area and delay of the Tie and Majority circuits which followed the use of these circuits is presented in Table 3.1.
This work was submitted to IEEE Transaction Material and Reliability [14].
To address the needs of the project the above will be implemented (with order RMC(3,6)) and referred to as RMC Extended. Furthermore it is feasible to apply an improvement to this method by reducing the size of the RMC matrix as in Section 2.7.8. When this improvement is made (again with RMC(3,6)) in the project, is will be called RMC Extended Reduced.

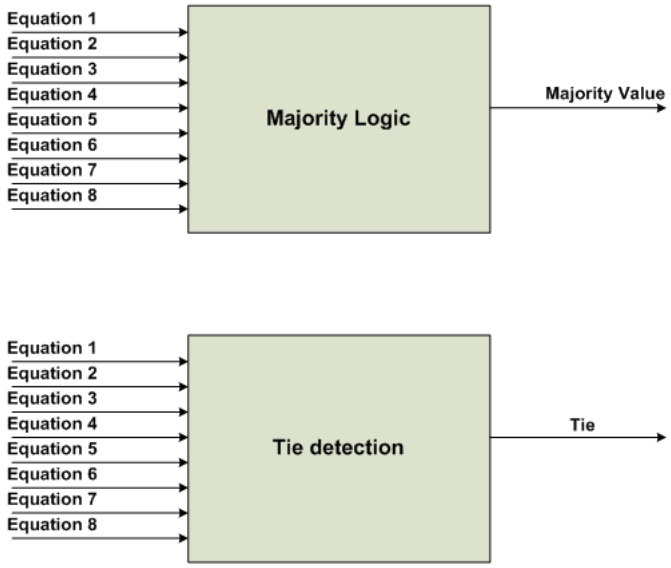


Figure 3.2: Illustration of the Majority and Tie circuit

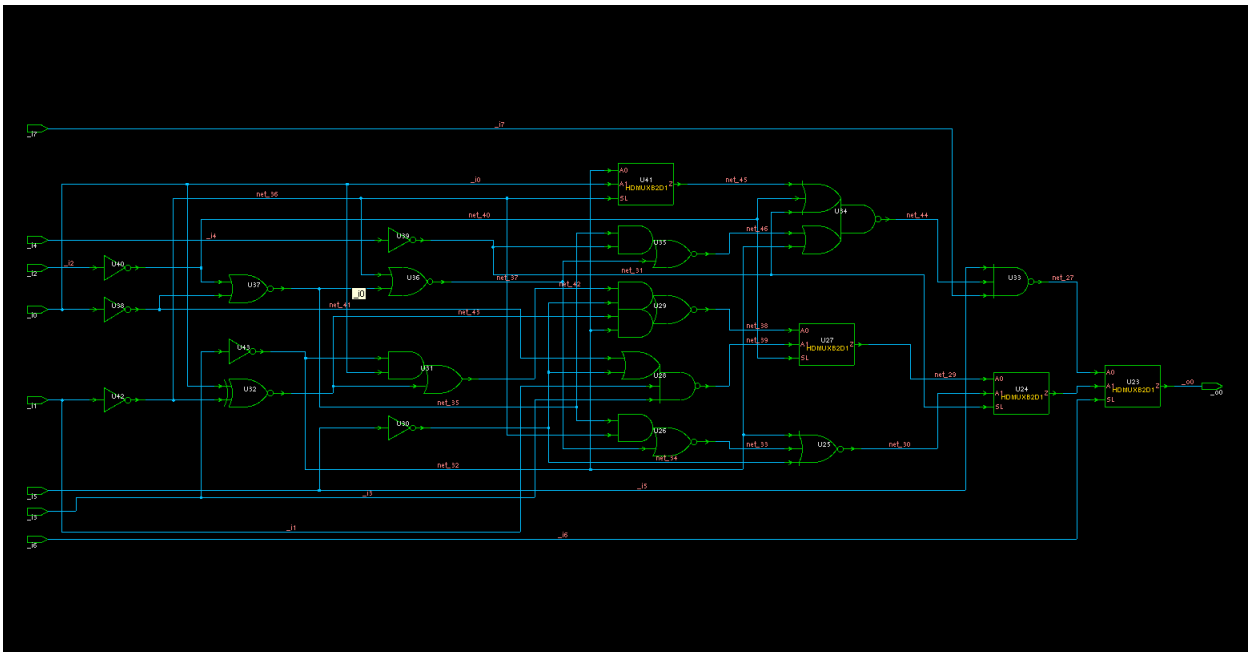


Figure 3.3: Schematic diagram of a Tie Circuit

All the possible combinations were examined (by the use of Matlab fault model) and the results shows that the new method is success in all possible situations.

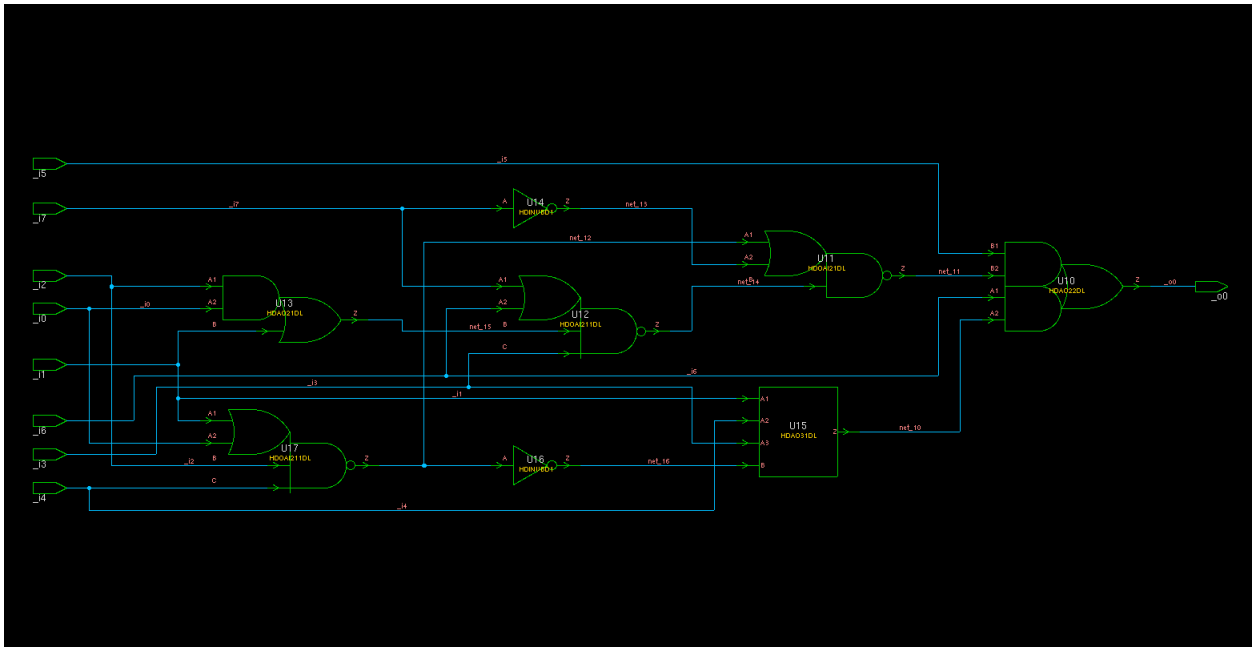


Figure 3.4: Schematic diagram of a Majority Circuit

Table 3.1: Area - Delay Informations of Majority and Tie Circuit

Circuit	Area($\mu.m^2$)	Delay(ns)
Majority Logic	116.1	0.59
Tie Logic	400.0	0.59

3.2 Overhead - Cost

For each fault tolerant method a significant factor which contributed to its effectiveness is the overhead. The overhead is the penalty needing to be paid to convert the standard memory design to a design with fault tolerant abilities. The hardware circuit was implemented using VHDL language and the characteristics were measured with Synopsis tool using 180nm technology. Through the use of the Synopsis tool the author received measures according to area, power and delay overheads. A new measure was defined to represent the overhead penalty for each method which was called cost. The cost is calculated by multiplying the 3 overhead factors: area, power and delay.

3.3 Mean Time To Failure (MTTF) - Mean Time To Catastrophe (MTTC)

Two new metrics are believed necessary to manage the results. The first metric was called catastrophe and the second, failure. The fail situation is described as the situation in which a method, reaches its maximum ability to correct or to detect an error and will be able to give a fail signal when another error appears. The catastrophe metric is defined as the situation in which a technique has exceeded its maximum capabilities and in the event of an error it will give wrong (catastrophic) results, transparent to the user. It is obvious (according to the Sections 2.7.3,2.7.4 and 2.7.5) that the methods which are able to support the failure situation are the parity code method, the Hamming code, the RMC Extended code and the RMC Extended Reduced code. On the other hand the catastrophe situation can appear in all the fault tolerance techniques which take place in comparison except the Parity Check code.

Faults models for each one of the methods in C language were created. In the simulation the target was to explore the behaviour of each technique for the presence of errors and to find their durability against these errors. To increase the accuracy of the simulation 1 million samples were used as well as a high performance computer (HPC) named Bluecrystal.

Through the use of the fault model the Mean error to Failure (METF) and the Mean error to catastrophe (METC) were measured in order to compare methods. The algorithm of the METF it is present at the Figure 3.5.

When the algorithm is star is initialise the variable which is use. The variable which is interesting is the variable called TRY (as is shown to the step two). At the third step of the algorithm adjust randomly error to the chip and increase the TRY variable by one. At the fourth step is check if at the errors row exist, other error. If this happened then reduce the ability of the fault tolerant method (which is used) by one and move to the step six. Otherwise is going back to the third step to adjust another error to the design. In the case which the algorithm is been to the step six then is check if the method which is use is reach the maximum capabilities to detect errors. If this is true then the algorithm is moving to the seventh step in which calculate the METF. The METF is the value of the TRY variable

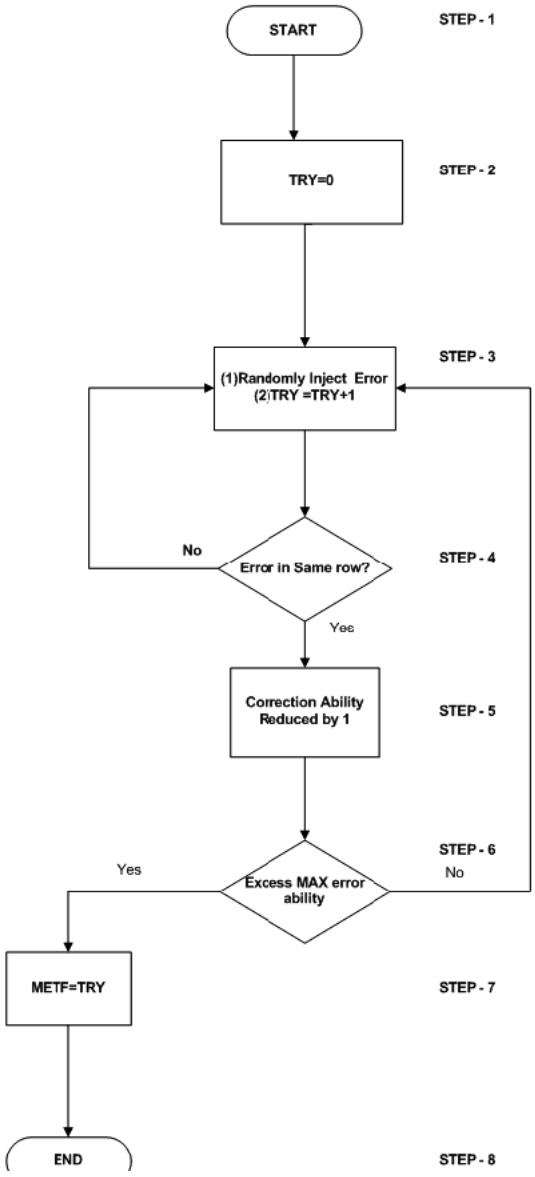


Figure 3.5: Algorithm of METF

The METC algorithm is presented to the Figure 3.6.
The same algorithm is used to calculate the METC. The only different is in the step six in which the algorithm check if the numbers of error per row is reach the maximum capability to correct error (instead to detect as is happened in the METF algorithm).
Moreover with the use of equation 3.1 it was possible to calculate the MTTC and by replacing the METC with the METF it was possible to calculate the MTTF as presented in Equation 3.2). The MTTF and MTTC can be calculated as the METF or METC respectively divided by the multiplication of the memory size with the fault rate. The memory size which was tested was 32KB and the fault rate $1X10^{-6}$ errors per day.

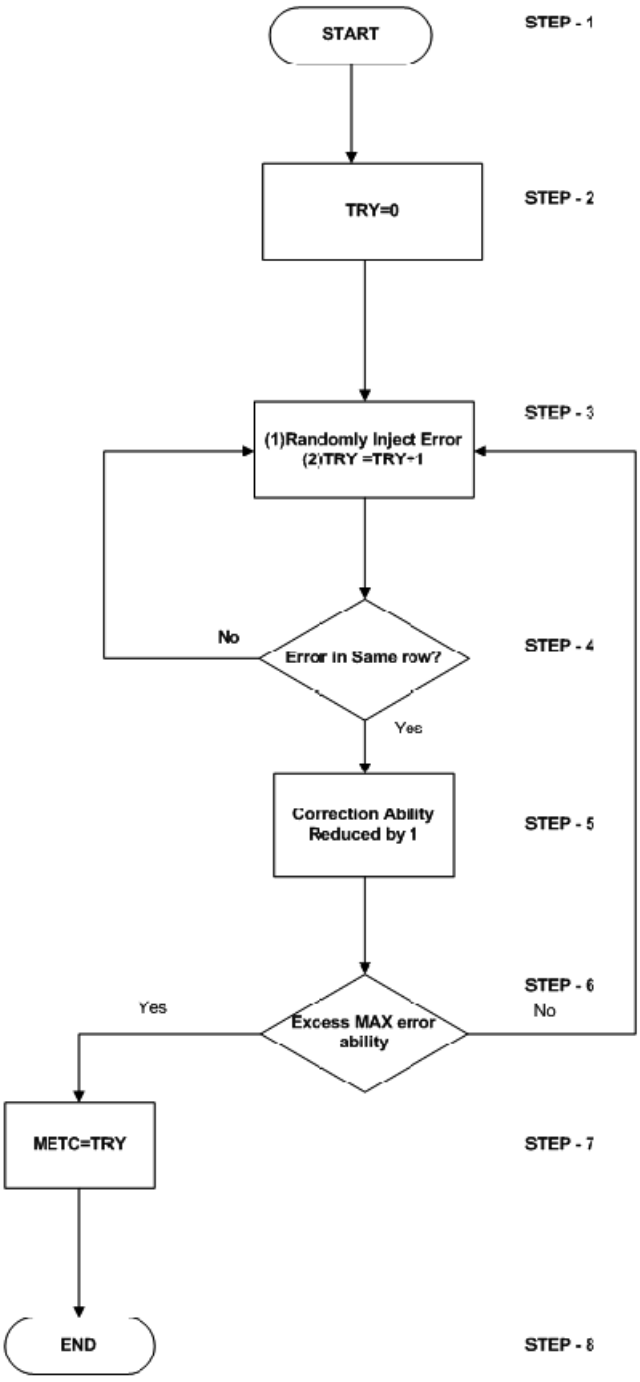


Figure 3.6: Algorithm of METC

$$MTTC = \frac{METC}{MemorySize.FaultRate(\lambda)} \tag{3.1}$$

$$MTTF = \frac{METF}{MemorySize.FaultRate(\lambda)} \tag{3.2}$$

It needs to be noted that the Cost of each technique is a reasonable high. The MTTF and MTTC values are given in terms of Cost. This is possible if the MTTF and the MTTC are divided by the Cost.

3.4 Yield Improvement

As it was mention in the introduction the second purpose of the project it was to investigation the usage of Fault tolerant techniques to improve the Yield. In the Sections 3.4.1 and 3.5 it will describe the two methods which was used to improve the Yield.

3.4.1 Yield Improvement by Fault Tolerant Technique

The second purpose of the project was to use the fault tolerant methods to improve the yield during the manufacturing process. During the manufacturing process the chips are created using a wafer. The problem which appeared was that the wafer suffered from defects. Defects are defined as the errors which subsist to the wafer (because of the process variation). The main objective in this Section is to use fault tolerant techniques as a "tool" to solve or at least to alleviate this problem. Any fault tolerant method has special abilities to detect or even to correct a different number of errors, so one objective was to check the effectiveness of the ability of these special methods to reduce the number of useless chips created by the wafer and increase the number of good chips.

To investigate the effect of the use of fault tolerant memories chips instead of standard memory chips, memory designs with embedded fault tolerant techniques were used. Many simulations were used with each one of the methods examined with the target to find the number of good chips for each method. Furthermore with these simulations the writer was able to compare the good chips in the presence of the different errors per wafer. More particularly it was able to simulate 2000, 3000, 5000 and 7000 errors (defects) per wafer.

For each one of the simulations it compared the number of chips where there were no errors per row, a maximum of 1 error per row, a maximum of 2 errors per row and a maximum of 3 errors per row. The reasons for comparing different cases is to examine the effects of each method compared to the numbers of good chips in combination with the reliability skills for each case. For instance a chip which is made with the RMC code and has 3 errors is a good chip because the RMC code is able to correct 3 errors. On the other hand a chip which is made with a Hamming code and has more than 1 error per row is a defective and useless chip. Furthermore the chips which are made with the Parity check code and DWC code are not able to correct any bits (so they are not able to improve the Yield). For these two codes the only good chips, are the non error chips. The reason is that these codes are not able to correct any error (only to detect it). An important point to mention is that when an error correcting method is used to improve the yield its fault tolerance abilities are reduced. Despite the fact that the use of fault tolerant techniques will increase the number of good chips they also raise the size of each chip so it will reduce the number of chips as shown in Figure 3.7.

Table 3.2: Maximum Number of Chips

Method	Number of chips
Hamming Code	820
Parity Code	969
TMR	333
DWC	500
RMC Code	500
RMC Reduced Code	592
RMC Extended Code	500
RMC Extended Reduced Code	592

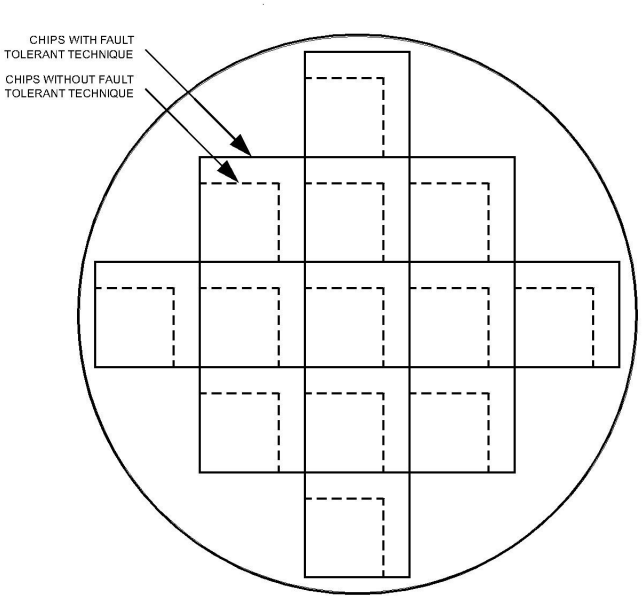


Figure 3.7: Example of chips overhead at wafer

It is important to give some information according the simulation. The simulation was constructed using C language with High performance computer (HPC) Bluecrystal.In the simulation, the maximum number of chips (without any fault tolerant techniques) which it is feasible to fabricate per wafer are equal to 1000. Through the use of fault tolerant techniques, because of the area overhead the number of the chips are reduced and vary depending on the technique. Table 3.2 presents the maximum feasible number which can be fabricated for each method. Also significant is that the errors at the wafer during the simulation follow a random probability distribution. In this way the simulation is more accurate to simulate the error behaviour of the fabrication process.

The algorithm which was used to simulate the Yield (for 3000 errors per wafer) is presented to Figure 3.8.The Yield algorithm after is start at the second step is calculate the number of chips which are feasible to implemented. According that we assumed that with the wafer is able to fabricate 1000 chips 32KB data memory without any correction method abilities the fabrication of chips with correction codes abilities will be able to create less chips. At the third step is defined the number of errors which it will simulate. During the step fourth adjust randomly error to the wafer. At the step five is calculate the number of chips which are examine (the No error per row, maximum 1 error per row, maximum 2 errors per row and maximum 3 error per row). At the step six is calculate the number of useful chips (which is dependent by correction method which is used).The next step is reduce the number of errors. At the eighth step is check if there is other errors to inject and if is true is go to the step four otherwise end the algorithm.

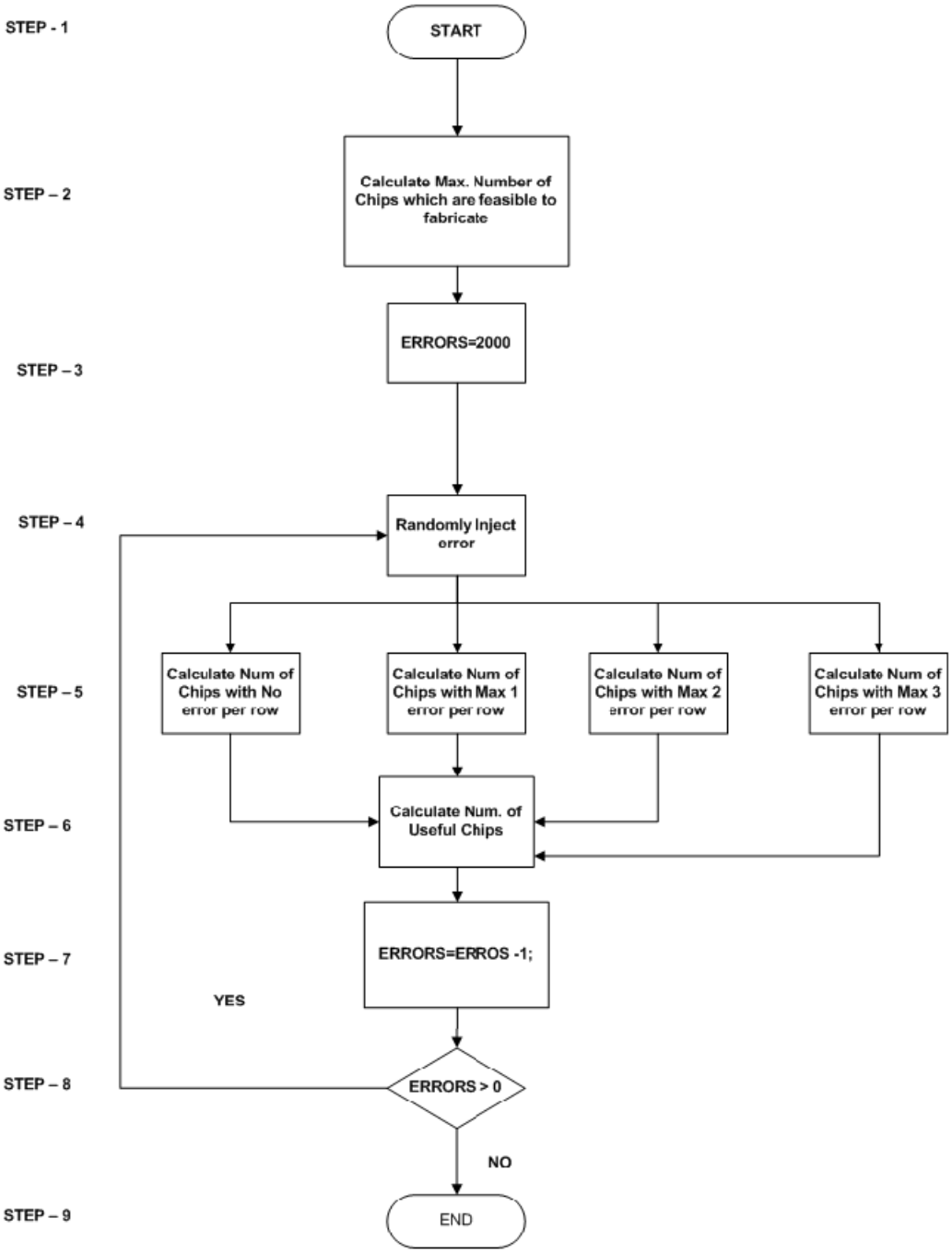


Figure 3.8: Yield Improvement with Error Correction Codes

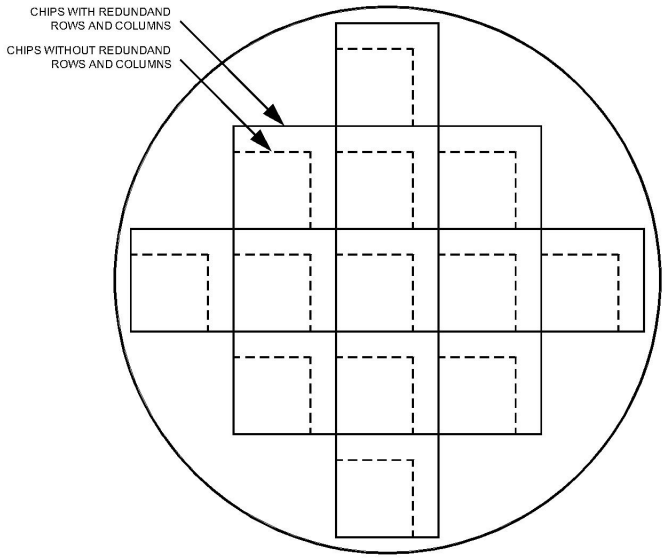


Figure 3.9: Overhead by the use additional rows and columns

3.5 Improve Yield by the use of redundant Columns(AC) and Rows(AR)

To further examine ways of improving the yield, another method was used. In this method is used the fault tolerant techniques (as the previous) but also is used additional rows and additional columns for each chip. During the fabrication process extra columns and extra rows are created in each chip. When errors appear to the chip the faulty row or column was removed and its position taken by the extra column or row respectively. By including this improvement to the simulation, a higher number of chips was able to be used (chips that were ready to use with no errors or if there were errors they were tolerated by the embedded techniques). However the construct of each chip with extra rows and extra columns, decreases the number of chips which are able to be made per wafer as Figure 3.9 shows. The simulation was created using C language following the same model as the yield simulation. Again the maximum number of chips to fabricate (without the fault tolerant method) was 1000 and with the extra columns and rows it is further reduced. The maximum number of chips fabricated for each method using redundant columns and rows is presented in Table 3.3.
At this section is used again the algorithm in Figure 3.8. The only difference appear at second step in which the calculation of the number of chips which are feasible to implemented is different because of the size of the chips are bigger (because of the extra rows and columns).

Table 3.3: Maximum number of tables

Methods	Addtional Rows(AR) - Columns(AC)	Number of chips
Hamming Code	AC=1 AR=1	800
	AC=1 AR=2	800
	AC=2 AR=1	780
	AC=2 AR=2	780
Parity bit code	AC=1 AR=1	941
	AC=1 AR=2	941
	AC=2 AR=1	914
	AC=2 AR=2	914
TMR	AC=1 AR=1	329
	AC=1 AR=2	329
	AC=2 AR=1	326
	AC=2 AR=2	326
DWC	AC=1 AR=1	492
	AC=1 AR=2	492
	AC=2 AR=1	484
	AC=2 AR=2	484
RMC	AC=1 AR=1	492
	AC=1 AR=2	492
	AC=2 AR=1	484
	AC=2 AR=2	484
RMC Reduced	AC=1 AR=1	581
	AC=1 AR=2	581
	AC=2 AR=1	571
	AC=2 AR=2	571
RMC Extended	AC=1 AR=1	492
	AC=1 AR=2	492
	AC=2 AR=1	484
	AC=2 AR=2	484
RMC Extended Reduced	AC=1 AR=1	581
	AC=1 AR=2	581
	AC=2 AR=1	571
	AC=2 AR=2	571

3.6 Benefit metric - Benefit based on the technique metric

Another two new metrics were introduced to compare the value of each one of the methods. The new metrics are called the Benefit metric and the Benefit based on the technique metric.

3.7 Benefit metric

An improved examination of the comparison between different methods was essential to involve the metric MTTC and the contribution to the yield for each method. Furthermore the overhead which was included in the fabrication cost of each method are factors which contribute to the metric. In this case each fault tolerant method keeps its reliability (to correct or detect) without the ability to increase the yield. The equation to calculate the benefit metric for each method is presented in Equation 3.3.

$$Benefit = \frac{MTTC}{fabrication\ cost} \tag{3.3}$$

3.8 Benefit based on the technique metric

The second metric which introduced called benefit based on technique. Despite the fact that the benefit base on technique measure is use again the equations 15 and 16 as the first metric, the difference is the way that the fabrication cost is calculated. For this metric, the useful chips (at the fabrication cost equation) are calculated according the ability of each technique to improve the yield. More particularly the useful chips are the chips with 1 error before the catastrophe (or fail for the case of parity check code).

The second metric which introduced is called the benefit based on the technique metric. Despite the fact that the benefit based on the technique measure is used again, equations 3.3 as the first metric, the difference is the way that the fabrication cost is calculated. For this metric, the useful chips (at the fabrication cost equation) are calculated according to the ability of each technique to improve the yield. More particularly the useful chips are the chips with 1 error before the catastrophe (or failure in the case of the parity check code).

The number of errors for each fault tolerant method used, are presented in Table 3.4. For example, with the use of the RMC code the useful chips for the calculation of the fabrication cost are the chips with no errors, maximum one error and maximum two errors.

Table 3.4: Error per technique at simulation

Method	Number of errors
Hamming Code	1
parity Check code	0
TMR	0
DWC	0
RMC Code	2
RMC Reduced Code	2
RMC Extended Code	3
RMC Extended Reduced Code	3

Chapter 4

Experimental Results Analysis

In this chapter we analysed the results which are received by the simulation. More particularly it was analysed the MTTF and MTTC, the yield and the Benefit metric.

4.1 Analysis of Overhead - Cost

As mentioned in Section 3.2 after the VHDL implementation and the use of the Synopsis tool it was possible to receive measures such as to area, power and delay. By multiplying these measures the Cost metric can be calculated. The values for each technique are shown in Table 4.1. Note that despite the fact that individual factors (area, power, delay) have units, the multiplication of the three factors is not supported with units. The reason is that the cost is used as a metric to compare the techniques which are divided by the same units. The simulation results are shown in Table 4.1. According to the experimental results the highest Cost value is provided by the TMR method. This is expected, having in mind that for the implementation of this technique it is necessary that the design is copied to three identical modules. Also significant value of Cost provided by the RMC Extended and RMC code.Both these techniques are needed to construct the high dimension RMC matrix capable of encoding or decoding a message. Furthermore the RMC Extended uses not only the matrix but also the extra hardware (Tie circuit) as described in Section 3.2 and so the Cost for this method is higher than the RMC Code. The next important value of the Cost is the RMC Extended Reduced code. The reduction of the RMC matrix decreases the overhead and as a result the Cost metric but its value is still high. For the same reason the RMC Reduced technique presented the next higher value Cost metric. A lower Cost value but still significant value is apparent with the DWC in which the implementation requires the design to be copied into two modules. Finally, with the lower Cost, the Hamming code suffers (needs only 7 bits overhead for each 32 bits codeword) and parity check with only 1 bit for each 32 bits codeword.

Table 4.1: Test

Fault Tolerant Method	Area (μm^2)	Power (mw)	Delay (ns)	Cost ($Area \times Power \times Delay$)
Hamming Code	35.66	56.88	3.04	6177.60
parity Code	30.17	48.13	2.57	3742.55
TMR	264.60	595.80	5.50	67074.29
DWC	132.30	132.40	2.75	48170.79
RMC Code	160.83	172.21	3.24	89905.21
RMC Reduced Code	135.70	145.30	2.73	54004.03
RMC Extended Code	161.23	172.63	3.83	106770.09
RMC Extended Reduced Code	136.04	145.65	3.23	64134.38

4.2 Analysis of Mean Time To Failure (MTTF) - Mean Time To Catastrophe (MTTC)

The simulation results of MTTF per Cost are presented in Figure 4.1. According to the results the most durable technique is the RMC Extended Reduced code with 18.56 day (assuming fault rate in days). Despite the fact that it suffers from the highest Cost (compared to other methods), because of the large number of errors per row which it is able to detect, it enables this method to provide the highest value in this field. The second highest value for MTTF appears to be the parity check code at 14.17. This is not because of its ability to detect errors, which is comparatively poor, (only 1 error) but from its extremely low overhead. Moreover the RMC Extended has the next highest value of MTTF with 12.64. The RMC Extended Reduced code also suffers from high Cost but also has large error detection capabilities. Finally the Hamming Code has the lowest MTTF as a result of its comparatively limited capabilities.

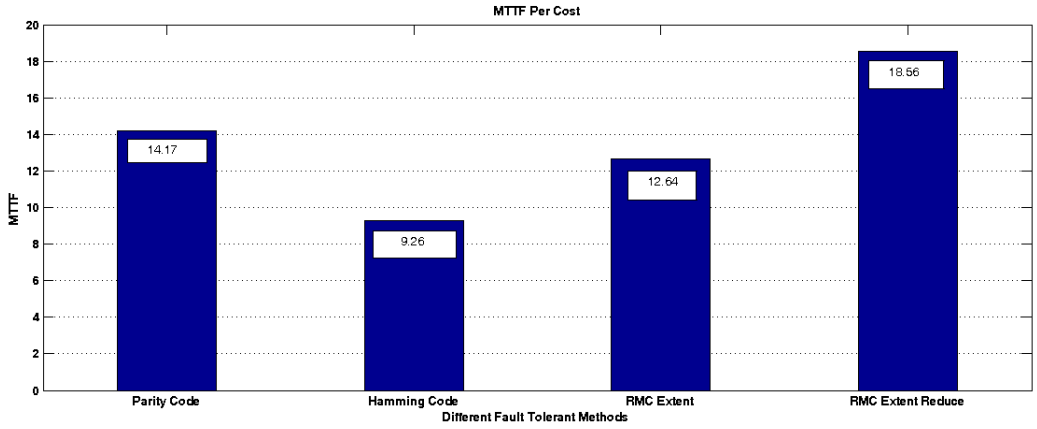


Figure 4.1: MTTF Results

At MTTC simulation the highest value appears at DWC with significantly less MTTC appearing with RMC Extended Reduced code. The other techniques have comparatively lower values. The results appear in Figure 4.2.

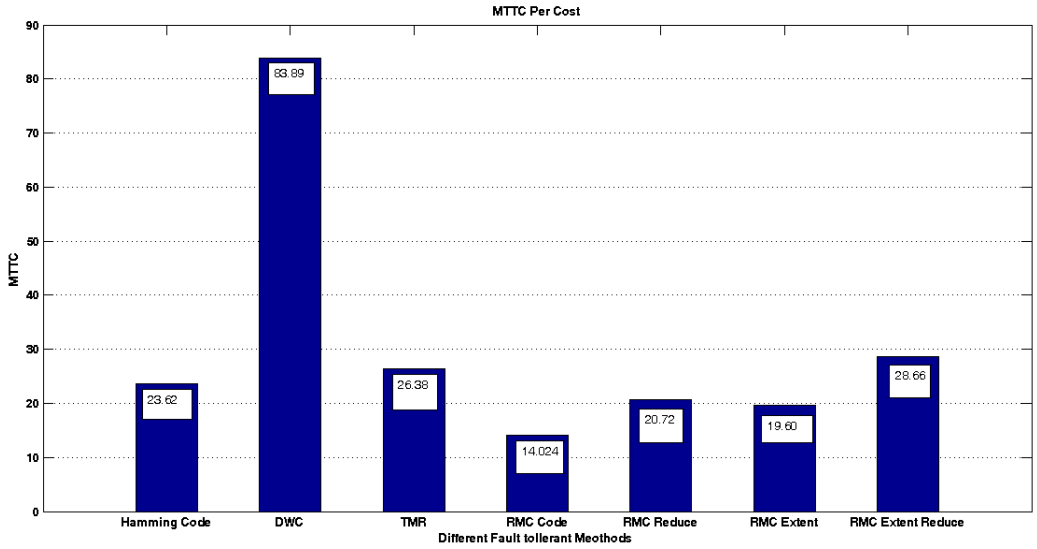


Figure 4.2: MTTC Results

4.3 Yield Improvement Analysis

4.3.1 Yield Improvement by the use of Fault-Tolerant methods

As described in Section 3.4 the purpose of this simulation is to use the Fault tolerant technique to improve the Yield. For an exact, successful conclusion to the behaviour of each fault tolerant method different numbers of errors per wafer were simulated (2000, 3000, 5000, 7000 and 9000 errors per wafer). An important mention is that when an error correcting method is used to improve the yield, its fault tolerance abilities are reduced to the chip. For example at 2000 errors per wafer the number of chips which can be made using each fault tolerant method are shown in Figure 4.4. The fault tolerant chips which are present still have the detection/correction abilities against errors after the fabrication process because the no error per row case was chosen. The no error case means that the good chips are chosen to be the chips which do not contain any errors.

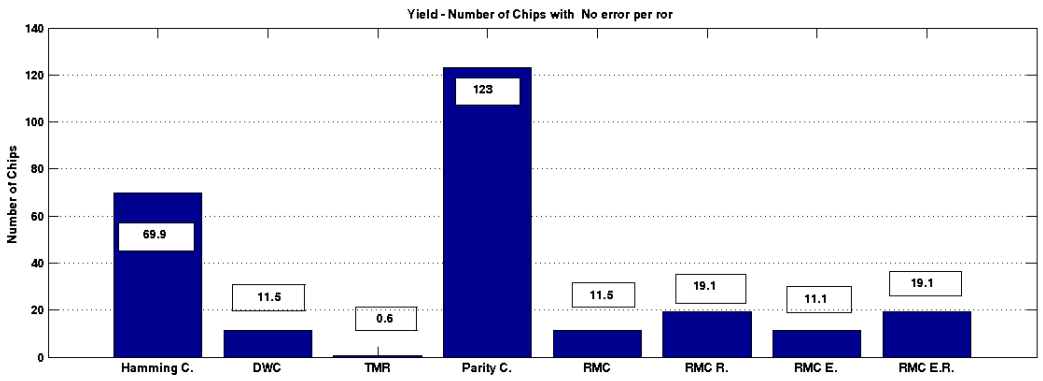


Figure 4.3: 2000 errors per wafer - Number of Chip with No error per row

Figure 4.4 presents the numbers of chips in which a maximum of 1 error per row (for the same number of errors per wafer) is present. The reliability of these chips, provided by the methods, is reduced to improve the yield. Using the Hamming code it is able to fabricate 764 chips because the Hamming code is able to correct to 1 bit error. Without using the Hamming code these 764 chips will not be able to be fabricated. The penalty paid for this is that these chips do not provide any fault tolerance to the future working life of these chips. In the same way, the use of the TMR code is able to fabricate 276 again with zero reliability. Moreover with the use of the RMC and the RMC Reduced code it can fabricate 445.6 and 534.9 respectively. The penalty in this case is that the reliability is reduced by 1 bit error to these fabricated numbers, so these 2 codes are able to correct 2 bit errors per row instead of 3 bit errors per row (which happens in the case of fabricated chips with 0 errors) . The same results using these 2 methods are repeated again for the RMC Extended and RMC Extended Reduced codes. The reason is that the RMC Extended and RMC Extended Reduced codes increase the ability of detection by 1 bit but do not make corrections. The detection error's ability cannot be used to improve the yield. For the same reason the number of fabricated chips, when the DWC or parity check methods are used, is zero. (Both of these methods are able to detect but not to correct any errors.)

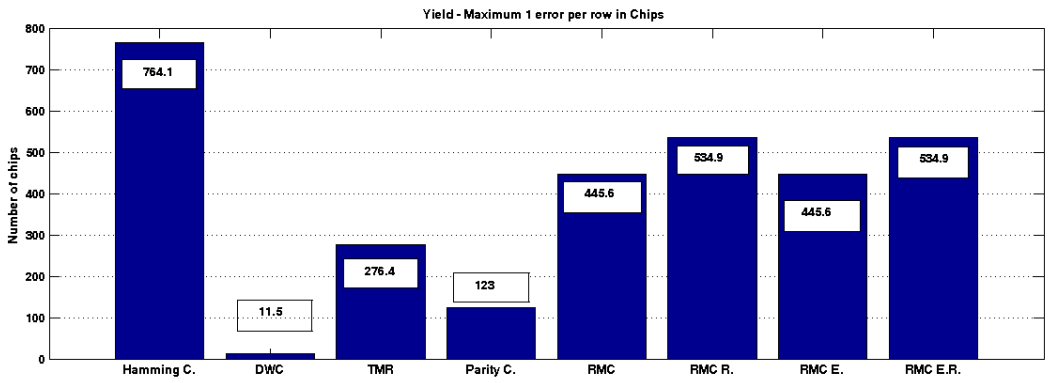
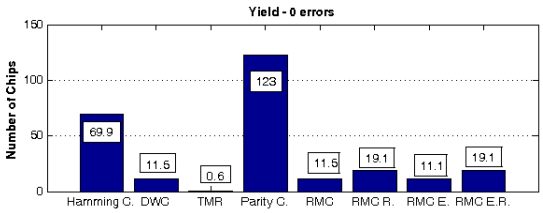


Figure 4.4: 2000 errors per wafer - Number of Chip with 1 error per row

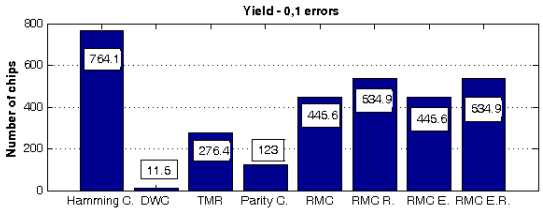
The 2 previous (Figure 4.4 Figure 4.4) graphs show the reduction of the reliability when the fault tolerant techniques are applied to improve the yield (important to note that all the results in Figure 4.4 would be less to zero if the fault tolerant techniques were not used).
It is an important note that the result of the simulation was not constructed in this way. Instead of measuring the number of chips independently in 4 cases (No errors, maximum 1 error per row, maximum 2 errors per row and maximum 3 errors per row) is added each time. For example, first, the number of no errors per row is simulated then the number of chips is simulated with a maximum of 1 error and this measure is calculated as per previous description . The reason for this is the need to examine the probability of improving the yield in each method. So the final results are calculated with no error, with maximum 1 error per row and maximum 2 errors row and are constructed in this way.
Figure 4.5 presents the 4 cases with the simulation results at 2000 errors per wafer. In the case of no errors per row the highest number of chips is given by the parity check code. This is not surprising because first, the number of errors per wafer is relatively low and as a result there are more chances for a high number of chips to exist with maximum 1 error per row. (In the case of no errors per row the parity check code is very powerful).Second, the parity check has extremely low overhead. In conjunction with these 2 parameters it is logical that the parity check has a high number of chips. It is worth noting the extremely low number of fabricated chips with the TMR code and finally the low numbers for the other techniques.

In the case of a maximum of 1 error per row the Hamming code produces a higher number of chips than the other methods. For the same reasons as for the case of no errors per row (high number of 0 and 1 errors per row and a low overhead) the results are expected. The ability to correct 1 error in conjunction with its low overhead puts the Hamming code with the highest number of chips. In this scenario the parity check code is moved from the first position to the penultimate position. The reason is its inability to correct errors. Furthermore, the other methods have a significant increase in their fabricated chips as a result of their ability to correct errors. The only opposite situation is the DWC in which its numbers remains stable (again because it cannot manage 1 error).
In the 2 last cases the number of chips for the RMC, RMC Reduced, RMC Extended and RMC Extended Reduced methods are raised (as a result of their ability to correct 3 errors) and the other methods remain stable. A major note is that despite the increase in the number

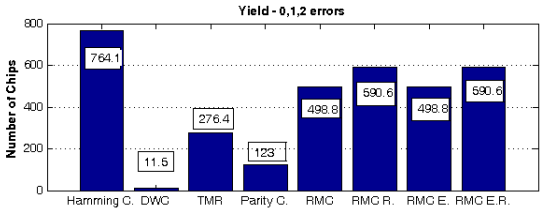
of chips which are created by the codes which are based on the RMC, they are incapable of reaching the value of the Hamming code. The reason, is the low number of errors per wafer and as a result the large number of 0 or 1 error per row.



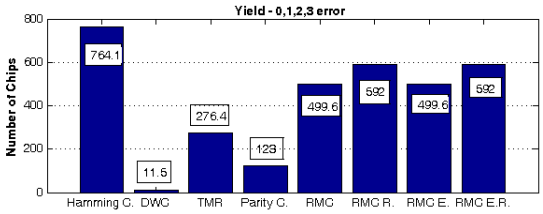
(a) No errors per row



(b) Maximum 1 error per row in Chips



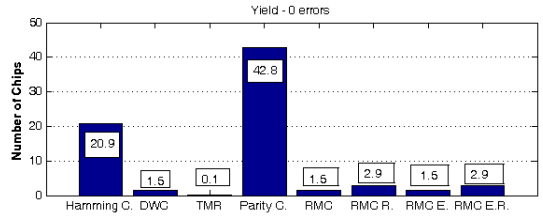
(c) Maximum 2 error per row in Chips



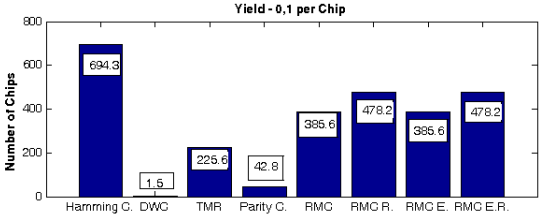
(d) Maximum 3 error per row in Chips

Figure 4.5: 2000 errors per wafer

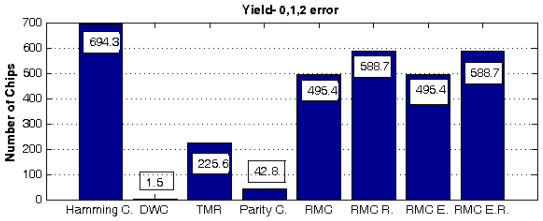
For the scenario of 3000 errors per wafer(Figure 4.6) the results follow the same behaviour as in the case of 2000 per wafer. In the case of no error per row, again the parity bit and in the case of 0 and 1 error the Hamming code has the highest values respectively. The only difference in comparison with the preview graph is the fact that the number of no error per row is significantly reduced which is a result of the increase of the errors per wafer. Another difference is that in the case of maximum 3 errors per row, the fabricate chips using the code which is based on RMC code are closer to the number of chips which are fabricate with the use of Hamming code (in comparison with the preview graph).



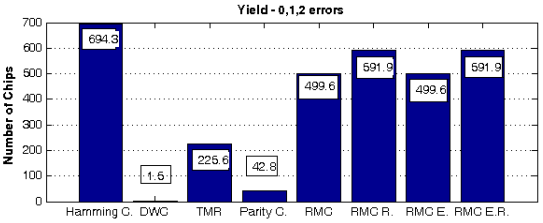
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chips



(c) Maximum 1 error per row in Chips

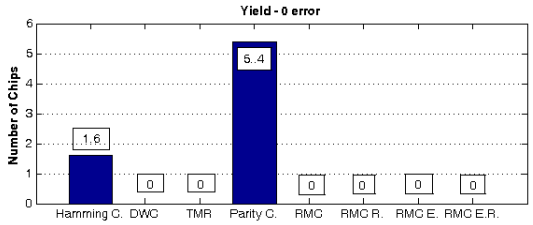


(d) Maximum 3 error per row in Chip

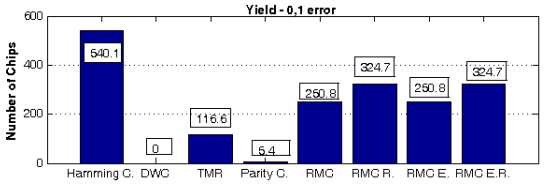
Figure 4.6: 3000 errors per wafer

The 2 previous scenarios. (With 2000 and 3000 errors per wafer it is safe to say that it does not respond to today's realities. The high exploration of CMOS technology and the high scaling of devices are the main factors creating numerous errors per wafer. These two sets of errors may be useful to simulate old-fashioned technologies but not with a modern design).

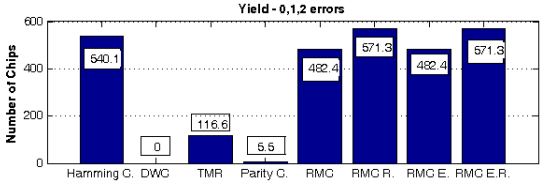
In the scenario of 5000 errors per wafer the results are evident in Figure 4.7. It is obvious that the number of no errors per row continues to decrease but the chips with a maximum of 2 errors per row and a maximum of 3 errors per row the errors increase. Furthermore, in the case of no errors the chips for almost all methods are zero (except Hamming and parity bit which are dramatically small). The most important difference is the fact that in these cases the RMC code and the RMC Extended code are slightly lower than the Hamming code and the RMC Reduced and RMC Extended Reduced code are slightly higher than the Hamming code.



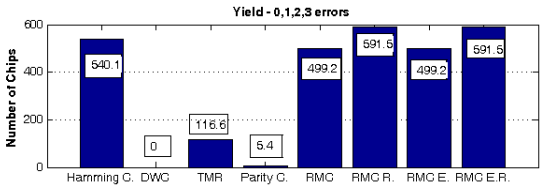
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chip



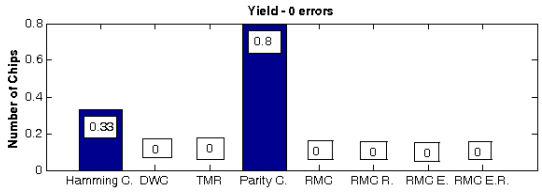
(c) Maximum 2 error per row in Chip



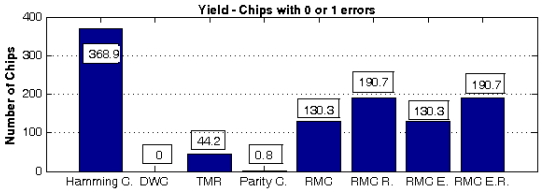
(d) Maximum 3 error per row in Chip

Figure 4.7: 5000 errors per wafer

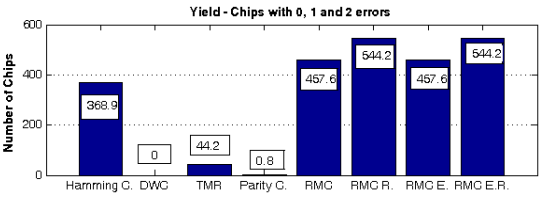
The results for 7000 errors per wafer are present to the Figure 4.8. The number of no error per row is still decreased and also the number of maximum 1 error per chip is decreased. On the other hand the number of maximum 2 errors per chip are significant raise. For one more time the hamming code is superior at the case of maximum 1 error per row. In the case of maximum 2 errors per row the codes the chips which are use methods which are based to RMC Code have higher number that the Hamming code. Moreover in these cases this difference is further increase. A notable mention is the superior of the RMC Reduced code and RMC Extended Reduced code to RMC and RMC extended code respectetevly.



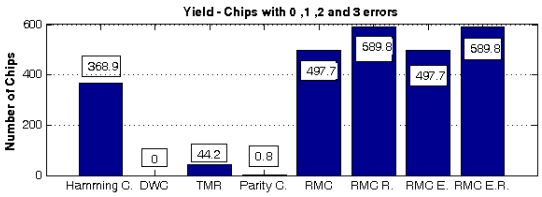
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chip



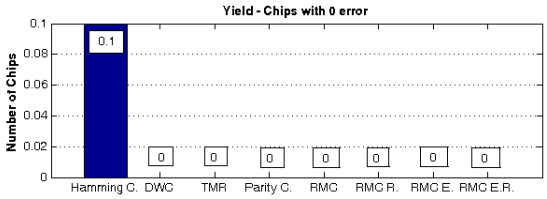
(c) Maximum 2 error per row in Chip



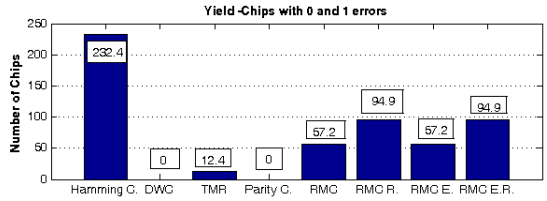
(d) Maximum 3 error per row in Chip

Figure 4.8: 7000 errors per wafer

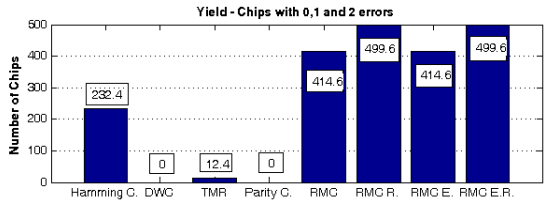
The final scenario it was to simulate 9000 errors per wafer(Figure 4.9). This quantity of errors is a pessimistic simulation. At the same model as all scenarios the Hamming code is very good against 0 or event 1 error per row. For more error per chip the RMC based codes are produce better results. More particular for maximum 2 errors per chip RMC and RMC Extended has little less than double number of fabricated chips than Hamming code. Moreover RMC Reduced and RMC Extended Reduced can produce more than double number of chips than Hamming code. In the case maximum 3 errors per row RMC Reduced RMC Extended Reduced are able to create more than 2.5 times the number of chips which are able to create by using the Hamming code. Also the difference between the chips of Hamming code and RMC and RMC Extended codes are increase.



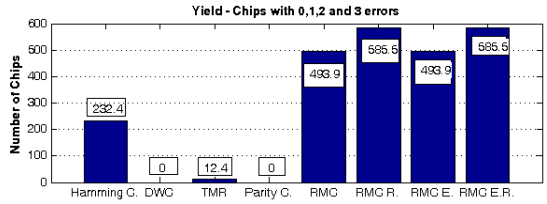
(a) No errors per Chip



(b) Maximum 1 error per row in Chip



(c) Maximum 2 error per row in Chip

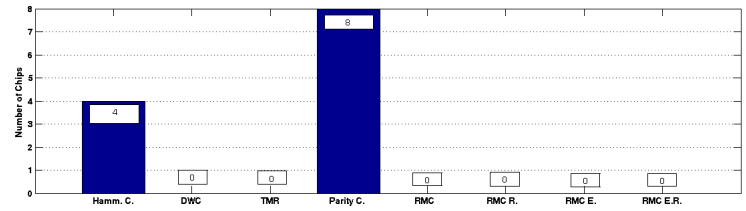


(d) Maximum 3 error per row in Chip

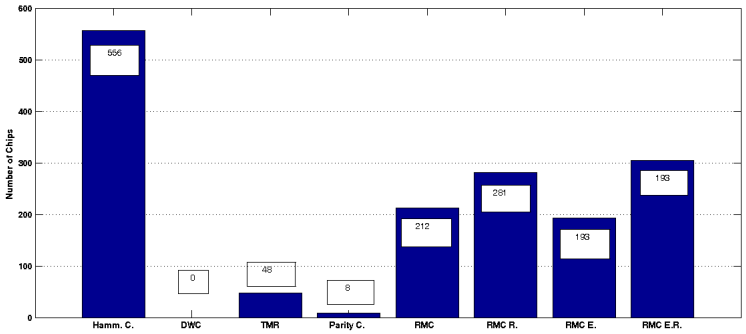
Figure 4.9: 9000 errors per wafer

4.3.2 Improve Yield by the use of redundant Columns(AC) and Rows(AR)

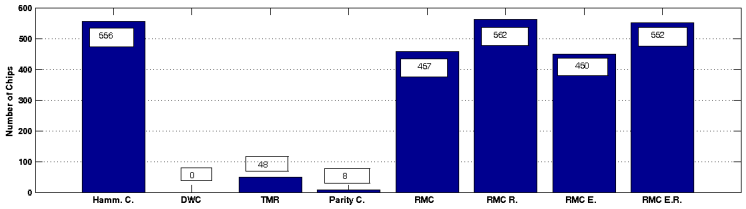
As it was described at the Subsection 3.5 a second effort to improve the Yield was establish. During this effort it was use redundant columns and rows to improve the Yield.
At the Figure 4.10 is presented the results at 4000 errors per wafer with 1 additional column and 1 additional row. According the 4000 errors per wafer scenario, the parity check code has the higher yield for the no error per chip. This is completely disappearing for the case of maximum 1 error per chip. In this case the highest number of chip is presented to Hamming code and the RMC's base codes are followed. In case of maximum 2 error per chip the most powerful method is the RMC Reduced code with 562 chips with slight smaller the Hamming Code at 556 and RMC Extended Reduced code at 552. At the last case the RMC Extended Reduced and RMC Reduced code appear to take the highest position with 581 and 577 chips respectively and third is the Hamming code. RMC and RMC Extended are followed. At all cases, the rest of the methods provide negligible number of chips.



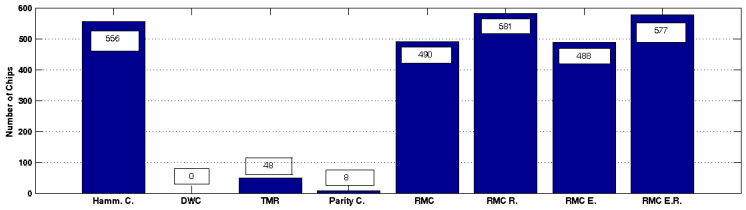
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chip



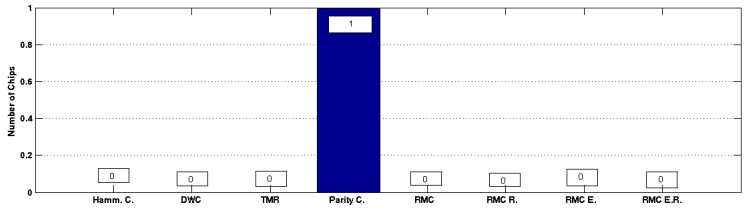
(c) Maximum 2 error per row in Chip



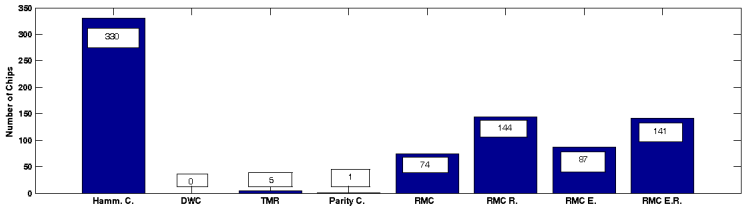
(d) Maximum 3 error per row in Chip

Figure 4.10: 4000 errors per wafer

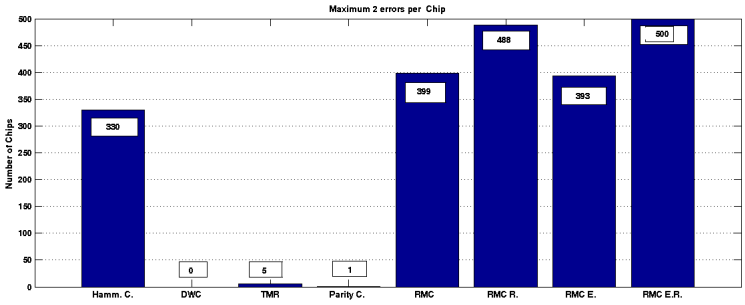
At the 6000 errors per wafer scenario(Figure 4.11) the results repeat the same behaviour like the previous scenario. The parity bit is the best way (even with only 1 chip) at case of no error and at the maximum 1 error per chips the best result appear at the use of Hamming codes technique at 330 chips . The difference is become visible at the case of maximum 2 errors per row in which all the methods which are based to RMC code has clearly better result that the Hamming Codes. This is more obvious at the last case in which the difference between hamming code and RMC's based code is significant increase. Again the remaining techniques are even zero or negligible to mention.



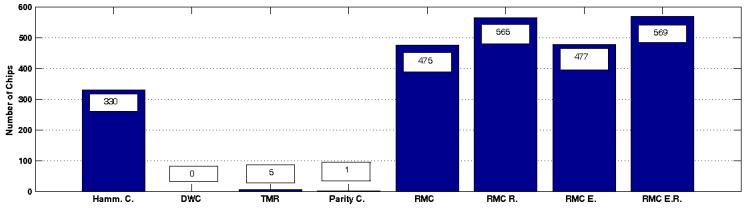
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chip



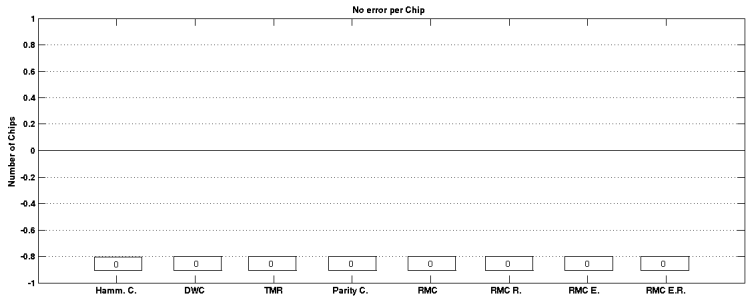
(c) Maximum 2 error per row in Chip



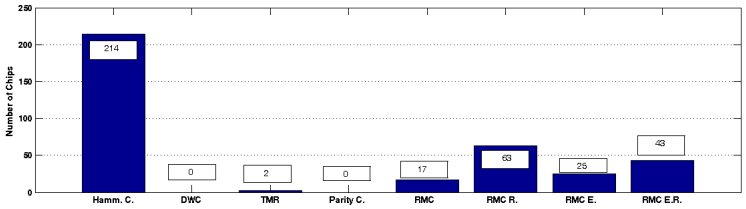
(d) Maximum 3 error per row in Chip

Figure 4.11: 6000 errors per wafer

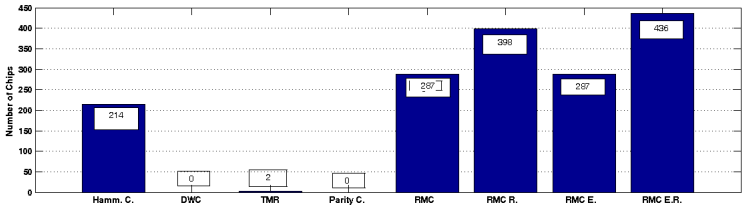
Finally at the last scenario of 8000 errors per wafer(Figure 4.12), for the case of no error per chips is equal to zero for all the techniques. This is expected because a high number error per wafer avoids the creation of 0 errors per chip. For one more time the Hamming code is present the highest number of chips at the case of maximum 1 error per row with 214 chips. At the case of maximum 1 error per row all the RMC's based code are higher than Hamming code and the same (with bigger difference) happen at the case of maximum 3 errors per row.



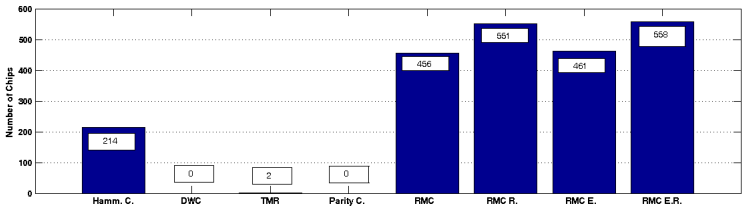
(a) No errors per row in Chip



(b) Maximum 1 error per row in Chip



(c) Maximum 2 error per row in Chip



(d) Maximum 3 error per row in Chip

Figure 4.12: 8000 errors per wafer

By the use of redundant columns and rows there is the advantage of replace a faulty columns or rows with faulty free columns or rows respectively. Despite this advantage the use of extra columns and rows reduce the feasible number of chips which are created. To summarise, by the use of additional columns and rows the increase of the chips number is the relative low.

4.3.3 Analysis of Benefit metric - Benefit based on the technique metric

During the need to establish metrics which are completely comparing the abilities of different techniques two new metric was constructed.

4.3.4 Analysis Benefit

According the experimental results, the highest value in each case is the parity bit code. This is expected because of the extremely low overhead of the parity bit code which tent to increase the number of chips which are feasible to fabricate. As a result of the significant number of chips which are fabricated, the fabrication cost reduced and this fact increase the benefit metric. Notable high values are present to the benefit metric of Hamming code again to all the cases of errors. The benefit metrics for all methods are reduces when the errors per wafer are increase. All the other fault tolerant method has poorly performance.The reason is the higher overhead for these methods, the number of chips which are feasible to fabricate is reduced, the fabrication cost is increase and final the benefit metric.dramatically decrease. When the number of errors per wafer is increase the benefit metric for all techniques are reduce The reason is the fact that for high value for wafer the chance for exist 0 error per row are decrease. Another important mention is that for 5000 and 7000 errors per wafer all methods (except parity bit code and Hamming code which are negligible) are equal to zero.

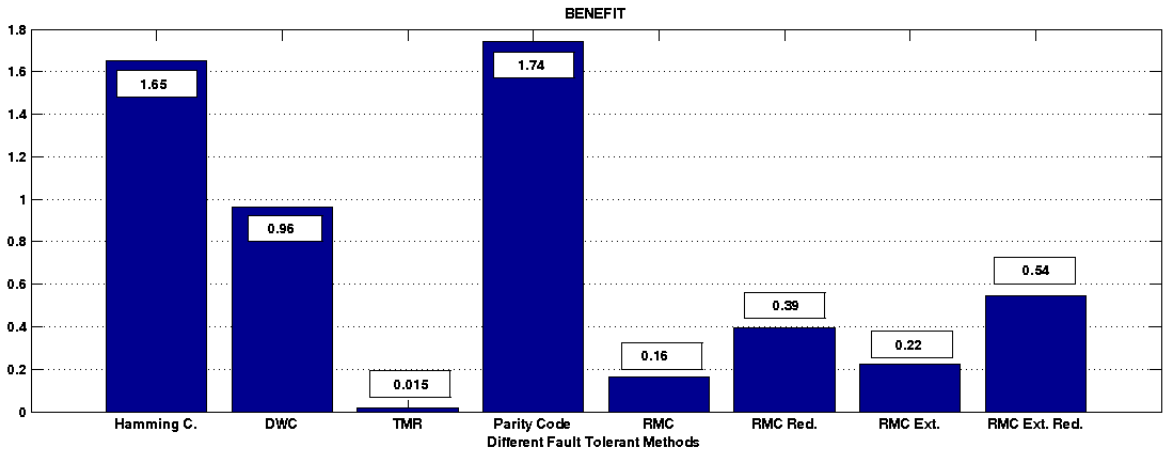
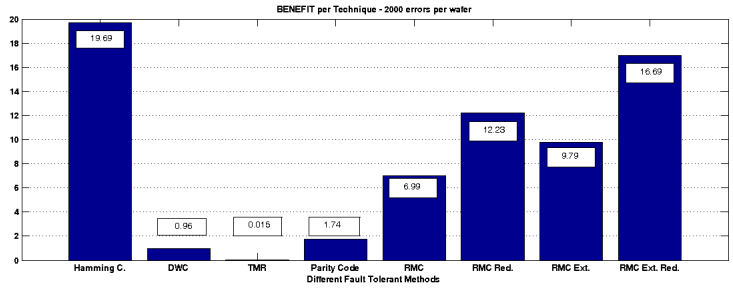


Figure 4.13: MTTF

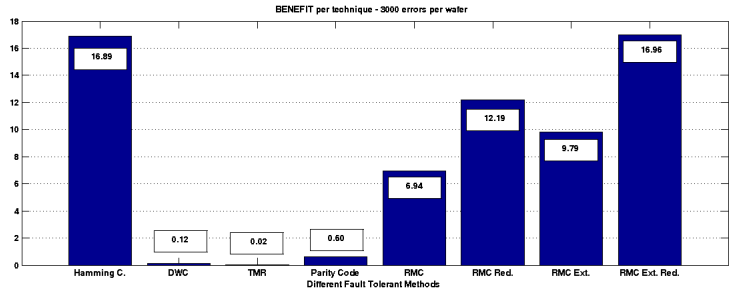
4.3.5 Analysis of Benefit metric based on the technique metric

According the simulation results for the metric benefit base to technique in the case of 2000 errors per wafer is showing in (Figure at 4.14.Hamming Code has the maximum performance.The codes which are base to RMC follow the Hamming code in terms of performance. Particulary the RMC extended Reduced code for 2000 errors per wafer is has small distance from Hamming code with value of 16,96 .The values of parity bit , DCW and TMR codes are dramatically small. At the 3000 error per wafer is the maximum value is present at the RMC Extended Reduced code with 16.96 which is slightly bigger that the Hamming code value of 16.89. Furthermore the values of RMC, RMC Reduced and RMC extended is also significant to the 6.94, 12.19 and 9.79 respectively. At the scenario of 5000 errors per wafer the maximum measure is present at RMC Extended Reduced code with 16.95 which is significant higher

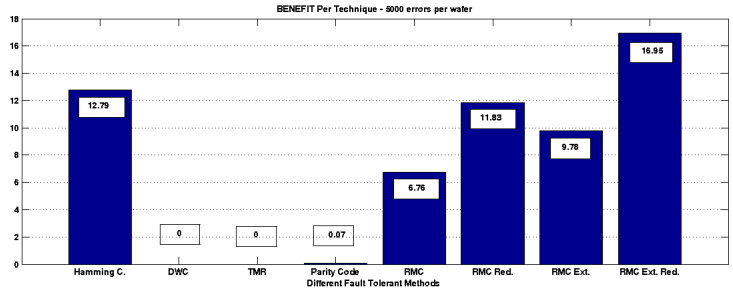
from the second hamming code at 12.79. Also important high is the value of RMC Reduced at 11.83. The RMC and RMC Extended are follows with 6.76 and 9.78 respectively. The DWC and TMR techniques are equal to zero and the parity check with extremely low value to 0.07. Move to the scenario of 7000 error per wafer the RMC Extended Reduced code at 16.9. The hamming code has a considerable reduction at 8.72.Also significant are the values of RMC, RMC Extended and RMC Reduced with 6.41, 9.75 and 11.27. The rest techniques are either zero or negligible.
As author argue at yield Section the simulation at 2000 and 3000 per wafer simulation is not represent a new technology devices. Because of CMOS scaling the new technologies suffer by higher number of errors per wafer at fabrication process. As a result the 5000 and 7000 error per wafer, are nearest to nowadays reality.



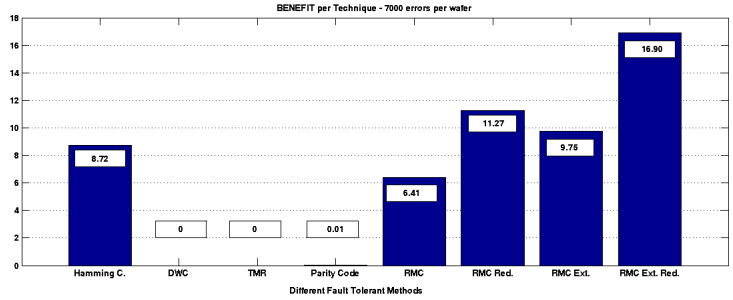
(a) Benefit - 2000 errors per wafer



(b) Benefit - 3000 errors per wafer



(c) Benefit - 5000 errors per wafer



(d) Benefit - 7000 errors per wafer

Figure 4.14: Simulation Results for different errors per wafer

The next 9000 error per wafer simulation is a pessimistic scenario. As Figure 4.15 shows the only notable values are the values of the code which are base on RMC. The higher value is present to RMC Extended Reduced with 16.78 and with considerable distance of the second RMC Reduced with 10.35. The others notable methods are RMC Extended at 9.68 and RMC with 5.81. An important mention is the dramatically reduce of the Hamming code at 0.002

and the remains techniques are equal to zero.

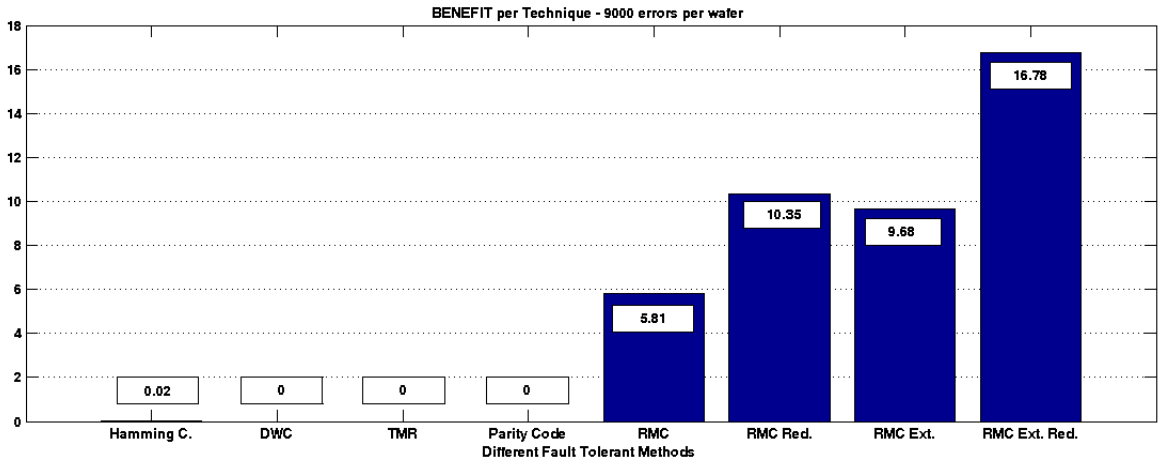


Figure 4.15: Benefit - 9000 errors per wafer

To summarise is obvious that considerable techniques according to the metric benefit base technique are RMC Extended Reduced, RMC Extended, RMC Reduced, RMC and Hamming code. The Hamming code is (my comment) low error wafer in which the codes which are base to RMC have significant values but less than hamming code. Particularly RMC Extended Reduced techniques is only slightly less from Hamming code at 2000 errors per wafer. When errors per wafer are increase the metric for Hamming code sinks and the RMC's base codes increase. In case of 7000 errors per wafer the value of RMC Extent Reduce is almost double from the value of Hamming code. Furthermore in the case of 9000 errors per wafer the Hamming code is around 0.002 and the same time the RMC Extended Reduced is 16.78. Moreover is clear that in the RMC's base codes in all the scenarios the RMC Extended Reduced present the higher values follow by the RMC Reduced, RMC Extended and finally by the RMC.

4.4 Comparison of Novel technique

A crucial parameter of the project was to compare the novel technique which was proposed with the old technique. This novel technique is based to an existing technique in which we have added an extra circuit to extent its reliability capabilities. The new technique is called RMC Extent code and is an extension of RMC code technique. When the reduction property of the classic RMC is applied to the new technique then the technique is call RMC Extent Reduce code. At this Section we try an exhaustive comparison of the new technique with the existing techniques.
Firstly is important to mention the abilities of the new method. The standard RMC code capability is to detect and correct 3 errors. When another error is appear this method is unable to correct or even to detect the error and the memory design (in which the fault tolerant method was apply) is going to catastrophe situation.(As explained at Section 4.2 the catastrophe situation is define as the situation in which a technique has exceeded its maximum

Table 4.2: Comparison MTTF-MTTC

Method	RMC Code	RMC Extended	Improv. (%)	RMC Reduced Code	RMC Extended Reduced Code	Improv. (%)
MTTC	14.024	19.6	28.44	20.72	28.66	27.7
MTTF	0	12.64	100	0	18.56	100

Table 4.3: Comparison -Fabrication Cost

	Error per Wafer	RMC	RMC Extended	Improv. (%)	RMC Reduced	RMC Extendend Reduced	Improv. (%)
Fab. Cost	2000	2.00	1.69	-15.54	2.00	1.68	-15.60
Fab. Cost	3000	2.01	1.69	-15.84	2.00	1.68	-15.59
Fab. Cost	5000	2.07	1.75	-15.56	2.00	1.69	-15.60
Fab. Cost	7000	2.18	1.83	-15.91	2.00	1.69	-15.61
Fab. Cost	9000	2.41	2.00	-17.01	2.02	1.70	-15.64

capabilities and in the event of an error it will give wrong (catastrophic results). At the new technique, its ability was improved so except the 3 error correction and detection is able to detect 4 errors. The first advantage of the new technique is that exceeds of the MTTC. Also the new technique (because of the ability to detect another error), provides a higher MTTF. The difference and the percentage improve of the new technique according to the standard method is presented in the Table 4.2.

When the fault tolerant techniques are use to improve the Yield the new technique and the standard RMC has the same abilities. The reason is the fact that the new technique is able to detect 1 more error but is unable to correct more errors. The ability to detect error is useless to improvement of the Yield. The improvement of yield is feasible when the correction capabilities of fault tolerant techniques are use to repair the defects which are appear at the manufacture process. When the new technique is not provide more errors correction abilities then is not expected to be more efficient to the Yield improvement.
Another major parameter of the new method is the overhead which is follow by the use of the new method. The new technique is base to the standard RMC code with the addition of extra hardware. The addition of extra hardware, increase the overhead of the design and also raise the fabrication cost for this method. The fabrication cost is defined as the 1000 divided by the useful chips. The useful chips are the chips which are feasible to fabricate from the wafer. The increase of the overhead, reduce the number of the useful chips and with this way decrease the fabrication cost. The fabrication cost for different number of errors per chip is present to the Table 4.3.

Finally according the Benefit base technique metric, in which involve all parametric, author compared the new technique. The Table 4.4 show the difference and the percentage improvement according benefit based on technique metric.

Table 4.4: Comparison - Benefit

	Errors per wafer	RMC	RMC Extended	Improve(%)	RMC Reduced	RMC Extended Reduced	Improve(%)
Benefit	2000	2.00	1.69	-15.54	2.00	1.68	-15.60
Benefit	3000	6.94	12.19	75.62	9.79	16.96	73.23
Benefit	5000	6.76	11.83	75.02	9.78	16.95	73.26
Benefit	7000	6.41	11.27	75.75	9.75	16.90	73.28
Benefit	9000	5.81	10.35	78.08	9.68	16.78	73.34

Chapter 5

Conclusion - Critical evaluation

In this chapter a critical evaluation of the project is provided. The aims and objectives have been completed successfully. Moreover, a step by step analysis of different stages of the project has been carefully analysed in line with aims and objectives.

First, a complete individual analysis of all the existing techniques is reviewed. The analysis was made according to the functionality of each method, how it worked and what the achievements and limitations are with regard to error detection and correction.

During the second task a novel technique was constructed which was based on an existing technique. The novel technique (which for the needs of the project was called RMC Extended) exceeded the limitations of the existing method. An extensive analysis of the functionality of the new method and the achievements are presented. Also the limitations and its penalties are listed.

In task 3 a VHDL model of each one of the techniques was implemented. With the use of these models and using the Synopsis tool it was able to analysed the overhead of each method. From these measures it was possible to calculate the new metric measure Cost. As the result shows, the fault tolerant technique with the highest Cost is the TMR method. RMC Extended and RMC's codes also have significant overhead but lower than TMR. Significantly, lower Cost values are present in the RMC Reduced, RMC Reduced Extend and DWC. Finally a lower Cost value appeared in the Hamming code and the lowest in the parity check Code.

During the 4th task, after constructing the fault model of each method and defining the MTTF and MTTC metrics, it was possible to calculate these metrics. From these metrics it was possible to extract useful conclusions about the durability of each technique. The simulation result showed that the highest value of MTTF was provided by the RMC Extended Reduced code. The second highest value was the parity check code and the lowest the RMC Extended and Hamming code. The result of the MTTC showed that the highest value as derived from the DWC method with RMC Extended Reduced being the next highest.

The next target examined was the use of fault tolerant techniques to improve the yield. This part of the project examined the correction of errors and the effectiveness of each technique. When these fault tolerant methods are applied to each chip fabricated, the errors (defects)

can be corrected. With the correction of errors during the fabrication process the yield is improved but there is a reduction in the reliability. Again the simulation results extract from extensive C language models. Different numbers of errors per wafer were examined during the simulations and for each number per wafer 4 cases were checked. The cases which were tests were the no error per row, maximum 1 error per row, maximum 2 errors per row and maximum 3 errors per row. In each simulation the number of chips which can be fabricated are calculated. From the results it is obvious that the Hamming code is the most powerful method for improving the Yield for a small number of errors per wafer (2000 and 3000 errors) than the methods which are based on higher order RMC code. The DWC, TMR and parity check code have a very small impact on the improvement of the Yield. For 5000 errors per wafer the parity check code has the best improvement of Yield for no error per row and the Hamming code has the best improvement for a maximum of 1 error per chip. For a maximum of 2 errors per chip and a maximum of 3 errors per chip the most useful results were the codes which are based on the RMC code and the Hamming code. The RMC Extended Reduced and the RMC Reduced present the highest improvement to the Yield. In the case of 7000 and 9000 errors per wafer the no error case provides negligible numbers of chips for all the methods. For a maximum of 1 error per chip the highest number of chips was provided by the Hamming code. Where there is a maximum of 2 errors per chip and a maximum of 3 errors per chip the highest number of chips are produced using the RMC Extended Reduced and RMC Reduced code followed by the RMC and RMC Extended.

To conclude, the Hamming code is effective in improving the Yield for only a small number of errors per wafer. (As mention previously the small numbers of errors per wafer do not represent modern technologies but only older ones. Given higher errors per wafer, for maximum of 1 error per chip the Hamming results are higher than other methods. An exhaustive examination of the ability of each method to improve the yield showed that the RMC Extended Reduced and RMC Reduced have higher abilities to improve the Yield with the RMC Extended followed by the RMC Reduced. The remaining methods provided negligible number of useful chips.

The next task was to examine other uses of the fault tolerant methods to improve the Yield. Apart from the use of the fault tolerant techniques redundant rows and columns were also used in the fabrication process of each chip. The results were disappointing because the feasible number able to be fabricated was not a significant increase.

In Task 7 two new metrics were introduced. The first metric was called Benefit and it showed the capabilities of each fault tolerant method. It involved the MTTC of each method in conjunction with the fabrication Cost. The Benefit metric is calculated where there are no errors per chip so it does not contribute to the improvement of the Yield. From the simulation taken different number of errors per wafer and with the help of C language valuable results were extracted. The result of this metric showed that the parity check code provides higher values followed by the Hamming code. The remaining methods present significant lower values.

The second metric was called the Benefit based technique and is a metric which combines the MTTC of each technique in conjunction with its abilities to improve the Yield. The results analysis showed that for small numbers of errors per wafer the higher value was provided by the Hamming Code followed by the RMC code. For 3000 errors per wafer the Hamming

Code is equivalent to the codes based on RMC. With 5000 and 7000 errors per wafer the superior methods are the RMC Extended Reduced and RMC Reduced and the Hamming Code and the remaining methods based on the RMC code provide lower values. With 9000 errors per wafer the only valuable methods are the RMC codes with the highest being the RMC Extended Reduced code. For all the cases the TMR, parity check, and DWC methods provide insignificant values.

To conclude, with a high rate of errors per wafer the only valuable methods are the techniques which are based on the RMC code and the highest values are provided by the RMC Extended Reduced code. The Hamming code provides important values only for a small number of errors per wafer.

Finally, the proposed technique (RMC Extended,RMC Extended Reduced) was examined in comparison with existing methods. During the examination the new capabilities of the error detection and correction methods were explored. These were the MTTF - MTTC metrics, Fabrication Cost and the metric Benefit based on the technique. The RMC Extended compare to the RMC method improved its detection ability by 1 error but the correction abilities remain the same. The MTTC increased by 28.44% and the MTTF by 100% (the existing technique did not provide any MTTF values). The fabrication cost decreased from 15.54 to 17.01% (for 2000 per wafer). With this metric (Benefit based on technique) there is an improvement from 75.62% (for 3000 per wafer) to 75.75% (for 7000 per wafer). Similar improvements exist in the comparison between the RMC Extended Reduced and RMC Reduced.

5.0.1 Critical Evaluation

This research project entitled "A framework for evaluation of fault tolerant memories in advanced technologies" can be considered a success. The motivation behind the project was to evaluate different error correcting techniques in memory. Although there are a few occasions where we encountered difficulty however overall the results were encouraging. A complete design methodology for incorporating fault tolerance was developed. The project was successfully because at the end a complete framework of different methods was created. The next step is to integrated the proposed techniques in a real memory block design and evaluate the performance parameters such as read time , write time static noise margin. Further an efficient layout can be constructed including the error correction schemes.

5.0.2 Contribution

My contribution in project can be summarise as follow:

- In depth study of differents Fault tolerant methods.
- Construct the model of each method in Matlab to check their fun ctionality.
- **Create a novel Fault tolerant technique which is based on an existing technique. At this technique the ability against errors is excise the standard method. The novel technique was submitted to IEEE Transactions Device and material Reliability.**

- Create the model (in Matlab) of the new technique to check the functionality and the capabilities against errors of the new technique.
- Analysis of the functionality of the new method through the use of a model which was created in Matlab.
- Construct the techniques in VHDL language and used Synopsis tool to receive measures such as area, power and delay.With these measures it was able to calculate the Cost metrics.
- Construct the fault model for each technique using C language and used it to investigate the MTTF and MTTC metrics.
- Construct models in C language to investicate the effects of use fault tolerant techniques to improve the Yield.
- Construct the models in C language to investicate the effects of use redundant rows and columns in conjunction with fault tolerant techniques to improve the Yield.
- Construct models in C language to investigate the metrics Benefit and Benefit per technique.

Chapter 6

Future work

Fault tolerant techniques are a vital part of modern designs. The reason is the high scaling of Cmos technology which causes considerable soft errors. As a result, the creation of new techniques and the improvement of the existing techniques became an urgent part of the digital design.
As mentioned in the section on the RMC Extended a new technique was constructed during the project. This technique is proposed to be published in IEEE Transaction on Device and Materials reliability. The new technique was named RMC Extended (in this project) and it was based on an existing technique (RMC) which was modified by adding an extra circuit. With the extra circuit it was able to extend its capabilities and enable it to detect another extra bit.
Further work can complement this method to improve its capabilities against errors. With the right combination and the addition of an extra circuit it is possible to increase (except for the detection abilities), the correction abilities of the technique.
Moreover the extensive use of the network on chip (Noc) architectures in modern digital designs increases the needs for fault tolerance (as a result of the high rate of errors in these types of architectures). A case study comparing the effects when different fault tolerant techniques are used in this architecture type is crucial.
To summarise possible future work could be as follows:

- Further improvements of the new technique could be made with higher error correction capabilities.
- A case study comparison of fault tolerant techniques could be completed with Noc architectures.

Bibliography

[1] Costas Argyrides. *Novel fault-tolerant techniques for the improvement of reliability and yield in advanced technol*. PhD thesis, University of Bristol, 2008.

[2] Key J.D. (eds.) Assmus Jr., E.F. Designs and their codes. *Press Syndicate of the University of Cambridge, Cambridge*, pages 116–125, 1992.

[3] Robert C. Baumann. Soft errors in advanced semiconductor devices-part i: The three radiation sources. *IEEE Transactions on Device and Materials Reliability*, 1(1):17–22, 2001.

[4] Robert C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, 2005.

[5] Michael L. Bushnell and Vishwani D. Agrawal. *Essentials of Ellectronics Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Kluwer academic publishers.

[6] B. Cooke. Reed muller error correcting codes. *MIT Undergraduate Journal of Mathematics,*, 1999.

[7] Stephania Costas Argyrides and Dhiraj K. Pradham Loizidou. Area reliability trade-off improved reed muller. In *SAMOS VIII Workshop, SAMOS VIII*, 2008.

[8] Paul E. Dodd and Lloyd W. Massengill. Basic mechanisms and modelling of single-event upset in digital microelectronics. *IEEE Transactions on nuclear science*, 50(3):583–602, 2003.

[9] Christoforos N. Hadjicostis. *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*. Kluwer Academi, 2002.

[10] Masao Kasahara Kinichiroh Tokiwa, Tatsuo Sugimura and Toshihiko Namekawa. New decoding algorithm for reed-muller codes. *IEEE Transaction Information Theory*, 28(5):779–787, 1982.

[11] Shu Lin. *An introduction to Error-correcting Codes*. Prentice-Hall electrical Engineering series, 1970.

[12] M. Moris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Pearson Prentice Hall.

[13] Sandeep Gupta Niraj K. Jha. *Testing of Digital systems*. Cambridge University press.

[14] Costas Kokkinos Antonio Maestro Pedro Reviriego, Costas Argyrides and Dhiraj Pradhan. Decoding of reed-muller codes with additional error detection for memory applications. *IEEE Trans. on Device and Materials Reliability*, submitted.

[15] D. K Pradhan and S. M. Reddy. Error-control techniques for logic processors. *IEEE Transactions on Computers*, c-21(12):1331–1336, 1972.

[16] Dhiraj K. Pradhan and Nitin H Vaidya. Brief contributions roll-forward and rollback recovery: Performance-reliability trade-off. *IEEE Transactions on Computers*, 46(3):372–378, 1997.

[17] D.K. Pradhan, editor. *Fault-Tolerant Computing Theory and Techniques volume I.* Prentice-Hall, Englewood Cliffs, New Jersey,, 1986.

[18] Sebastian Raaphorst. Reed-muller codes. *Carleton University*, 2003.

[19] Lang Su. Fault simulation for stuck-open faults incmos combinational circuits. Master's thesis, Faculty of the College of Engineering and Technology Ohio University, 1993.