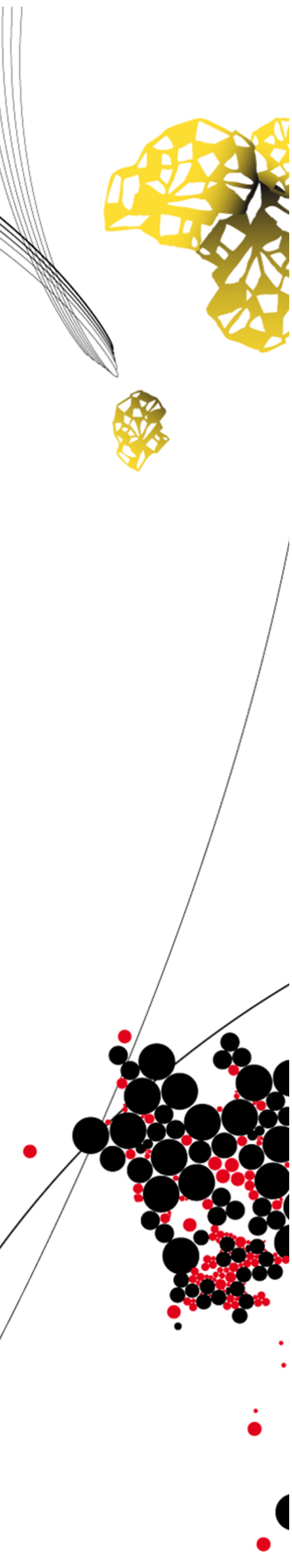


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Self-healing approximate multipliers in MAC

Vincent J. Smit

M.Sc. Thesis

November 13th 2020

Supervisors:
dr. ir. A. B. J. Kokkeler
dr. S. G. A. Gillani
dr.ir. M.S. Oude Alink

Computer Architectures and Embedded Systems
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Preface

This thesis was performed at the Computer Architectures and Embedded Systems group of the EEMCS faculty of the University of Twente in fulfillment of the requirements of acquiring a Master of Science degree in Computer Science, specifically in Embedded Systems.

Foreword

I found that during the master thesis completion process one tends to gain knowledge on three subjects. First of all you learn the subject material of your thesis. Secondly, you learn quite a bit about yourself and finally, oddly enough, how the game of cricket works.

For those readers whom are interested in the first of the three aforementioned learning points, I'd advise you to read the rest of this document. If you are more keen on learning about the final point, I can recommmend a conversation with my supervisor Mr. Gillani. In the unfortunate case that he is unavailabe, the Wikipedia page on the sport does a reasonable job as a substitute. This section however, I'd like to devote to the second point.

The people who followed me more closely during the time in which I was learning all this, know it was not always a smooth ride. Thankfully I did not have to endure everything alone.

First of all, I would like to thank the people at my volleyball club Harambee, especially those I got to spend my board year with, for sharing my passion for the greatest sport on the planet. I also got to share with you the love for the surprisingly pleasing color combination of orange and purple (and by extension electric yellow). I don't think there exists anything more dazzlingly magnificent.

Secondly to my friends and colleagues at 'De Beiaard'. I would like thank you all for both the theoretical and practical education regarding the production, distribution and consumption of a wide variety of alcoholic beverages. Here I would like to mention Louis van Appeven in particular, for his elaborate contributions in transforming my Denglish into English. His efforts will be rewarded with an honorary sixpack Kanon.

Finally, to my high school friends from Raalte. I'm glad we stuck together and I hope that we will keep finding each other accross the globe, as we always have.

Besides these groups of people I was lucky enough to have been a part of, I would also like to mention a few people in particular.

First of all I'd like to thank my supervisor Ghayoor. You have always been a positive force throughout the time I was working on my thesis. I enjoyed our conversations both on the subject matter as well as those regarding completely unrelated

issues like sports and languages. The phrase 'If you want to get rid of it, finish it' will stay with me for a while.

Secondly I'd like to thank my other supervisor, Andre. For your seemingly infinite patience, your sharp commentary on my work and for the fact that somehow there was always some room for me in your incredibly busy schedule.

Thirdly, I'd like to thank my parents for their contributions to my study in every way that they did. Even though we did not always agree on everything, I have always felt that I had your general support.

And finally my gratitude goes to my girlfriend Rosanne. Both in direct support, but also, and mabye even more so, in the confronting but honest concern regarding my progress and state of mind. I'm greatly looking forward to our next step together.

Concluding, I'd like to turn to mathematics to describe the process of writing my thesis and completing my degree accordingly.

$$x = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

$$y = \sum_{n=0}^{\infty} \frac{1}{n!}$$

$$process = \frac{1}{8} * \frac{1}{x} * y$$

It was a piece of cake. Almost.[1]

[1] For those reading whom are not so much into mathematics: x is an approximation of $\frac{1}{\pi}$. y describes Euler's number (e). Thus, $process = \frac{1}{8}* \sim \pi e$ ie. a part of something approximately pie.

Contents

List of acronyms

IC	integrated circuit
DSP	digital signal processing
POC	proof of concept
HA	half adder
FA	full adder
MAC	multiply accumulate
SAC	square accumulate
PPM	partial product matrix
ME	mean error
RMS	root mean square
MSE	mean square error
MED	mean error distance
SEC	static error correction
SECV	static error correction value
SH	self healing

Chapter 1

Introduction

Approximate computing is shown to be a promising strategy for developing a solution to combat the increasing energy consumption of computer chips. With this strategy an improvement on area and power performance is traded off against computational accuracy [1]. It has been demonstrated that a variety of algorithms and applications exists that can tolerate a non-accurate result. Examples of such applications include machine learning, data mining and image processing [1], [2]. Approximate computing exploits this tolerance to balance maximum power and area savings with minimal loss of accuracy [1]–[4].

For algorithms like addition [5], [6], multiplication [7], [8], multiply accumulate (MAC) [9] and square accumulate (SAC) [10] examples exist that show the feasibility of the application of approximation strategies. All of these design proposals use a variety of approximation techniques to find the optimal balance between saving power while introducing errors.

Error correction mechanisms are sometimes applied in order to reduce the error introduced by the original approximation. The downside of these mechanisms is that they tend to increase the area of the approximated design, and consume more power. A solution to add error correction without increasing the area was proposed in [10] for a SAC operation. This solution of self healing proposes the use of two parallel SAC units with inverse mean error. By summing the results of these parallel units, the error of the result of the complete operations is reduced.

1.1 Problem statement

In research, the approximate multiply accumulators are always equipped with some form of error correction. This correction is performed by either single multiplier designs with static error correction [9], or parallel designs that utilize self healing [11]. When self healing is applied to MAC, only absolute mirror multiplier designs ex-

ist. These absolute mirror designs include two parallel multipliers, which are each other's perfect opposite. For every inputs a and b, if one multiplier gives an error of $+x$, the other will give an error of $-x$. This is a nice property that assures that the error behaviour of both multipliers is similar. These absolute mirror multipliers are generally designed together. Designing an absolute mirror for any random approximate multiplier found in literature is not always straightforward. Moreover, sometimes these absolute mirror designs have even worse area or power characteristics.

A possible improvement can be made by combining two multipliers into a self-healing MAC design which are not each other's perfect mirror, but have some similar error properties. By exploiting some statistical and self healing properties of the iterative MAC algorithm, and with a large enough number of inputs, a design might be possible that is not absolute in mirroring each other's multiplicative behaviour. The idea is that those multipliers mirror each other's mean error and possibly other error characteristics. Therefore, when the number of inputs is large enough, the self healing MAC could still perform better than the absolute mirror designs. This type of multiplier pairs is referred to as mean error mirror multipliers. The central question this thesis attempts to answer will be, when considering parallel MAC structures, whether a self healing design utilizing the mean error mirror principle can perform better than similar designs using the existing strategies as mentioned before.

In this research, some absolute mirror multiplier MAC designs will be compared to a proposal for a mean error mirror MAC design. In this mean error MAC both multipliers have an equal but opposite mean error, but the two multipliers do not mirror each other's multiplicative behaviour exactly.

1.2 Contributions

This thesis contributes a proof of concept of a new self healing MAC design method that is an addition to the pareto optimal curve of existing self healing MAC designs. The applied method is the mean error mirror method for two multipliers. The main aim is a design which has a significantly smaller area, and a good area/error tradeoff compared to other design methodologies, given that the number of inputs is large enough.

1.3 Overview of document

This document first goes into the background of the field of research in chapter 2. In chapter 3, various options for multipliers suitable for approximate MAC are designed, simulated and implemented. These multipliers are then utilized in approximate MAC designs in chapter 4, where the results of four MAC design strategies are implemented and analyzed. This chapter also includes a proof of concept of the proposed mean error mirror design strategy. Finally, conclusions are drawn from these results in chapter 5.

Chapter 2

Background

In the research field of digital integrated circuit (IC) design, an increasingly relevant challenge is the reduction of energy consumption by logic circuits. Multiple strategies have been developed over time in order to meet this challenge. A promising and recently reappearing strategy is approximate computing. A simple idea where a reduction of energy consumption is traded off against a loss of computational accuracy in the algorithm that solves a particular problem. This approximate computing challenge entails two major subjects.

The first focuses on the applicability of approximate computing to certain algorithms. Not every algorithm is equally susceptible to the application of a form of an approximate approach. However, many applications are at least partially approximable without major impact on the final result of the performed task. Applicable fields are for example image processing, neural networking and data processing where the input data is subject to significant input noise. Identifying these partial structures of an algorithm has been subject of research in the past. A recent overview of several of these techniques is given in [2], but it is not further elaborated on in this thesis.

The second subject focuses on the methods to achieve such approximations. Approximate techniques exist at both software and hardware level. Since this research focusses on approximate MAC designs, the hardware level techniques are evaluated in greater detail in this thesis.

2.1 Terminology

Certain words and phrases will reappear in various places throughout this thesis. Some of these words can have an ambiguous meaning when left unexplained. Therefore, a few key terms are explained here, so that they can be clearly understood in the further reading of this document. The definitions are based on [11].

2.1.1 Quality

The term *quality* is referring to the error behaviour of a design. If a certain design A is of higher quality than a competing design B, this means that design A has a smaller error than design B for the specified error metric. Design A having a better *error behaviour* than design B has the same meaning as design A being of higher quality.

Accuracy and precision

When discussing the quality of a design, the two terms *accuracy* and *precision* are two ways to reason about error. Accuracy is the closeness of the measured values to a specific, predefined value. When reasoning on approximated circuit designs, the preferred value of error is generally zero. Therefore, an accurate design is a design that, on average, has an error near or around zero.

Precision means the closeness of the measurements to eachother. A precise design has all error values somewhat equal, where for a less precise design, the individual error values are more spread out.

2.1.2 Performance

All designs considered are evaluated for their quality(error) and cost(area). With all these designs, a tradeoff between quality and cost is presented. The tradeoff between these two metrics is also referred to as the *performance* of these designs. Therefore, when a certain design A performs better than a design B, design A yields a better tradefoff between the quality and cost factors.

2.2 Quality analysis

In any approximate circuit the output is subject to the error introduced in the design. The quality analysis that is performed to quantify this error behaviour utilizes the following basic settings.

2.2.1 Input distribution

The error behaviour of a circuit is strongly influenced by the distribution of the input data supplied to that circuit. Unless otherwise specified, in this thesis the range of the input values is defined by eight bits and the inputs are unsigned. The range of input values in base 10 is therefore any integer between 0 and 255. Throughout this

thesis, two probability distributions of the input values are considered. These are the uniform distribution and a normal or Gaussian distribution.

If the input distribution is uniform, every input value is equally likely. For a normal distribution the values around the mean value (μ) are more likely, whereas the more extreme values are much rarer. Whenever a normal distribution of inputs is mentioned, the following properties apply. The mean value (μ) = 127.5 and and the standard deviation (σ) = 42.5. This distribution covers μ +/- $3 * \sigma$, meaning that the entire range of values from 0 to 255 is covered. The distribution is also truncated at these values, as usually the normal distribution would continue indefinitely. The probability graphs corresponding to these distributions are given in Figure 2.1.

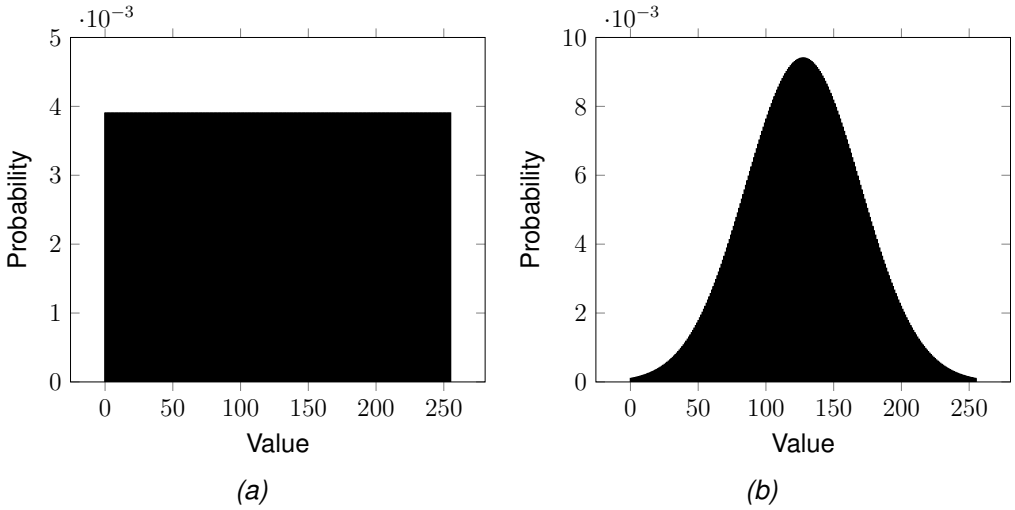


Figure 2.1: Input value probability for: (a) uniform input distribution (b) normal input distribution $(\mu = 127.5, \sigma = 42.5)$

2.2.2 Hardware function differences

Multipliers and multiply-accumulators are the two types of hardware functions that are subject to quality analysis in this thesis. Both are analyzed in a different manner.

Multipliers

The number of possible outcomes of an 8-bit multiplier is relatively small. Therefore, the entire solution space of an approximate multiplier can be analyzed completely in a result matrix and the deviation with respect to the correct solution can be presented in an error matrix. An example of the correct result matrix, an arbitrary approximate matrix and the corresponding error matrix is given in Table 2.1. The value of each cell is the result of the multiplication of its coordinates. Related to the error matrix, an error probability matrix can be calculated. To calculate the error probability matrix, the error matrix is multiplied elementwise with an input-probability matrix for either (uniform or normal) input distribution. Within that input-probability matrix the value in every cell equals the product of the probabilities of both of the inputs that form the cell's coordinates. With this method two error-probability matrices for each approximate multiplier are calculated. One for the case where the inputs are distributed normally, and one for uniformly distributed input. From the error and the error-probability matrices a variety of error metrics for this multiplier can be derived.

Multiply-accumulators

The multiply-accumulate operation is an iterative operation, as it sums all the results of individual multiplications together. This makes it difficult to generate a result and error matrix, as the result depends on the number of input pairs of which the product should be accumulated (shortly 'inputs'), instead of just the value of the inputs. Therefore, the multiply accumulators are analyzed by averaging a large number of accumulation results for specific input sizes.

2.3 Error metrics

The main concept of approximate computing is to reduce the power consumption and area requirements of a design. This reduction is traded off with a loss of accuracy in the result of the function that is performed by the design.

For a reduction in power consumption and area required for a design, some error can be tolerated in the approximate designs. When a larger power reduction is required, the error that has to be allowed is likely greater in return. Plotting the

a \ b	0	1	2	...	255
0	0	0	0	...	0
1	0	1	2	...	255
2	0	2	4	...	510
⋮	⋮	⋮	⋮	⋱	⋮
255	0	255	510	...	65025

(a)

a \ b	0	1	2	...	255
0	0	0	0	...	0
1	0	1	2	...	250
2	0	2	4	...	500
⋮	⋮	⋮	⋮	⋱	⋮
255	0	250	500	...	62500

(b)

a \ b	0	1	2	...	255
0	0	0	0	...	0
1	0	0	0	...	-5
2	0	0	0	...	-10
⋮	⋮	⋮	⋮	⋱	⋮
255	0	-5	-10	...	-2525

(c)

Table 2.1: Result matrices of (a) an accurate 8-bit multiplier, (b) some arbitrary approximate multiplier and (c) the corresponding error matrix ((b) - (a))

area gains against certain error metrics of a design, gives a good insight in the quality-cost tradeoff of a proposed design.

In determining the quality of a design, several metrics are used for analyzing error behaviour in approximate designs [11].

2.3.1 Error rate

Error rate, also referred to as error frequency, is the fraction of the incorrect outcomes over the total number of outcomes.

2.3.2 Error magnitude

Error magnitude refers to the numerical deviation of an approximation from the accurate result. This metric can be defined by various different values, which show statistical properties of the quality of a design. For each metric mentioned, the formulas for calculating them are given in de equations below.

First of all the mean error (ME) is an indication of the accuracy an individual operation. It is computed by summing all individual errors and dividing by the number

of values that were summed.

Also, methods for indicating the precision of the error are used. The values resulting from these methods are an indication of the difference between individual errors and the mean error of all errors. Examples of such methods are the root mean square (RMS) of the error and the mean error distance (MED) [12]. Also the mean square error (MSE) [11] is used in literature to indicate the precision of a design.

y_i = approximated result of the operation i

x_i = accurate result of the operation i

n = number of operations

$$ME = \frac{\sum_n^{i=1} y_i - x_i}{n}$$

$$MED = \frac{\sum_n^{i=1} abs(y_i - x_i)}{n}$$

$$MSE = \frac{\sum_n^{i=1} (y_i - x_i)^2}{n}$$

$$RMS = \sqrt{\frac{\sum_n^{i=1} (y_i - x_i)^2}{n}}$$

2.4 Approximate Computing Strategies on the Hardware Level

Various means to achieve the desired approximation are available, but the methods that will be discussed here are only concerned with approximations on the hardware level. While approximation strategies on the software or architecture level also exist, they are outside the scope of this research.

The strategies on the hardware level entail both approximations on the gate level and on the transistor level. On the gate level, the approximation of computation of occurs by removing gates from an accurate design in order to increase efficiency. On the transistor level the removal of transistors has a similar effect, but scaling the input voltage supplied to the circuit is also an option.

2.4.1 Voltage over-scaling

Voltage over-scaling entails lowering the voltage over the circuitry in order to put transistors out of order or in a slower operating mode. [13]

The idea behind this technique is to lower the voltage over a circuit to a value below a certain threshold which inherently decreases the power consumption of the

circuit. The negative side effect is that the behaviour of the individual transistors is influenced. With a lower than required voltage supplied, one of two things can happen. Firstly, the voltage over the circuit is too low compared to the threshold voltage of the transistor. Therefore, there will never be a current flow from the source to the drain, turning the transistor off. The second possible consequence is that a selected set of transistors is put in a slower operating mode. When these transistors are responding slower, they might become too slow and produce a result after the value is already read from the circuit. When individual gates are no longer operational, the switching activity of the transistors is reduced and therefore the design dissipates less power. However, timing errors are introduced because certain gates are operating too slow or not all, which makes errors imminent.

2.4.2 Approximate Adders

Gupta et al. [5] propose a method for designing an approximate full adder (FA) on the transistor level for digital signal processing (DSP) applications. By carefully removing transistors from an accurate mirror adder, three approximate adder designs are given. These approximations lead to a significant reduction of both the area requirement and the power consumption.

Another approach with transistor based adders is shown in [6]. In this case the accurate adder design is an accurate XOR based or XNOR based adder. The three approximated variants proposed show strong power reduction.

2.4.3 Approximate Multipliers

Various approximating techniques are applied to construct approximate multiplier designs. Examples of these techniques are truncation and the approximated addition tree.

A truncated design performs approximation by reducing the number of bits that is used in calculating the result. This can be applied to both the inputs and the partial product matrix (PPM) of a multiplier. An example of both of these truncation options is shown in Figure 2.2.

[8] applies the method of truncation, combined with some error correction features, in order to design multipliers with a low mean error. Due to the low mean error and low mean square error, these multipliers are shown to be feasible for use in MAC designs.

In [14] an approximate multiplier design is proposed that reduces the area of the multiplier by replacing a selected subset of the half adders in the addition tree with logic OR gates. This results in a signifcant reduction of area requirement for an 8-bit

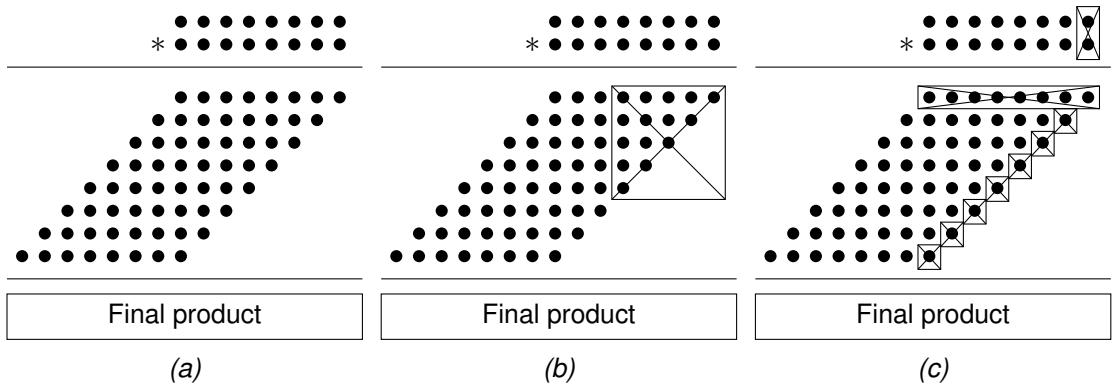


Figure 2.2: Truncation on an 8-bit multiplier: (a) A normal partial product matrix (b) Truncated PPM with all partial products in the five least significant bits removed (c) The least significant input bit removed.

multiplier.

The [15] paper presents a new technique to design signed and unsigned truncated multipliers. Simple formulas are developed in the paper to describe the truncated multiplier with minimum mean square error.

Another multiplier with approximation introduced in the addition tree is proposed by [16]. They employ a new approximate adder that limits its carry propagation to the nearest neighbours. The error recovery strategy that is added to the multiplier can be configured, so different levels of accuracy can be achieved.

Finally, [7] proposes constructing large multipliers with smaller ones. This means that an *n*-bit multiplier is constructed using four *n/2* multipliers. These smaller multipliers are then approximated. In this paper they present 4-bit, 8-bit and 16-bit multipliers that are constructed using 2-bit multipliers, where they introduce approximation in a subset of these smaller 2-bit multipliers.

2.4.4 Approximate Multiply Accumulators

As mentioned in the previous section, [8] proposes a number of multiplier designs for application in a multiply accumulate (MAC) structure. These multipliers are suitable for MAC application due to their low mean error and low mean square error.

A full example of an approximate MAC is proposed in [9]. In this paper a MAC is constructed with an approximate multiplier, using a combination of the multiplier of [14] and introducing some truncation to this multiplier. Static error compensation, as explained in the next section, is applied to reduce the magnitude of the error.

2.5 Error correction

Extra circuitry can be added to an approximate design to improve on the error introduced by the approximating components, while keeping in mind that the goal is still to use less area than the accurate designs. Two methods for compensating for errors found in literature are evaluated, static error correction (SEC) [9] and self healing (SH) [11].

2.5.1 Static error correction

Static error correction is a method for compensating for the error of an approximated design, by adding the mean error of that design to the result of every approximated result. This approach is depicted in Figure 2.3. The figure shows the multiplier M1 with input values a and b. The result of this multiplication yields some error ϵ_{a*b}. The mean error of this multiplier, ϵ_{mean_M1}, is then added to the multiplication result. The average error $\epsilon_{average}$ of the complete design should therefore be approximately zero.

This approach to error compensation is also applicable in multiply accumulate architectures. For example, consider a MAC that deploys the same multiplier M1 as in figure 2.3. The static error correction value (SECV) in the resulting MAC will be equal to $\epsilon_{mean_M1} * l$, with l the number of values to be accumulated. Over a large number of inputs, the mean error of the MAC with error compensation should thus approach zero. The method of static error addition for MAC is applied in [9]. The downside to static error compensation is the requirement for additional hardware to implement the compensating circuitry.

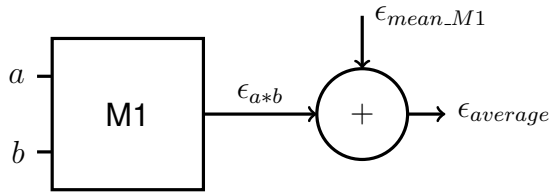


Figure 2.3: Multiply structure with static error correction

2.5.2 Self healing

When a design is based on an iterative algorithm and the operation is parallelizable, the self healing approach to error correction is an option. An implementation of this error correction technique can be found in [10], applied to a square accumulate architecture. In this paper, the self healing square accumulate structure that is proposed has a better quality output than conventional approximate computing

methodology. This technique is also applicable in MAC architectures by using two approximate multipliers. These two multipliers M1 and M2 have an inverse mean error ϵ. The accumulation stage acts as the self healing step, where the results from M1 and M2 are added together and their individual error should cancel out.

Figure 2.4 shows this approach. The multipliers M1 and M2 in this figure have an equal, but opposite error ϵ. The individual results are added together, before the values are accumulated. The resulting error ϵ_{mac} of MAC circuit should therefore approach zero.

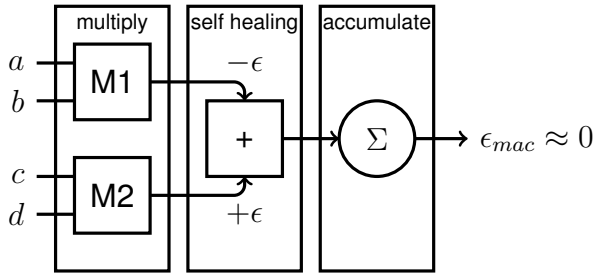


Figure 2.4: Self healing MAC structure with multipliers M1 and M2, having inverse error ϵ

Chapter 3

Approximate multipliers for self healing MAC

In order to apply the self healing strategy to a MAC structure as in figure 2.4, we need to find a pair of compatible multipliers M1 and M2. Compatibility in this sense means the error behaviour of the multipliers is equal but opposite, meaning that at least the mean error $\epsilon_{M1} = -\epsilon_{M2}$. The method for building self healing structures in hardware is conventionally with absolute mirror pairs, as utilized by [10]. An absolute mirror pair of multipliers in an approximate MAC is achieved when M1 and M2 are perfect opposites. This means that for every pair of inputs a, b, the result of $a * b$ with multiplier M1 will generate a result with an error magnitude of $-\epsilon$. The result of the same multiplication using multiplier M2 will give a result with error $+\epsilon$. The error is thus not only equal but opposite on average, but for every case.

3.1 Absolute mirror multipliers

In literature [11], examples exist for an absolute mirror pair using a recursive multiplier as shown in figure 3.1. In this multiplier, an 8-bit multiplier is constructed using 2-bit multiplier components. These 2-bit components are approximated. In figure 3.2, the 2-bit multipliers that are proposed are shown, with their corresponding results in table 3.1.

Both approximate 2-bit multipliers have exaclty one error case, when both inputs are 3 (or 11 in binary). In case of M1 in figure 3.2b, the error magnitude is -2 and the error of M2 in figure 3.2c is $+2$. Besides this perfect opposite error behaviour, both designs are also smaller in area compared to an accurrate 2-bit multiplier, as shown in table 3.2. Therefore, they are good candidate designs for application in absolute mirror multipliers.

The next task is finding the best designs of these 8-bit multipliers, which have the

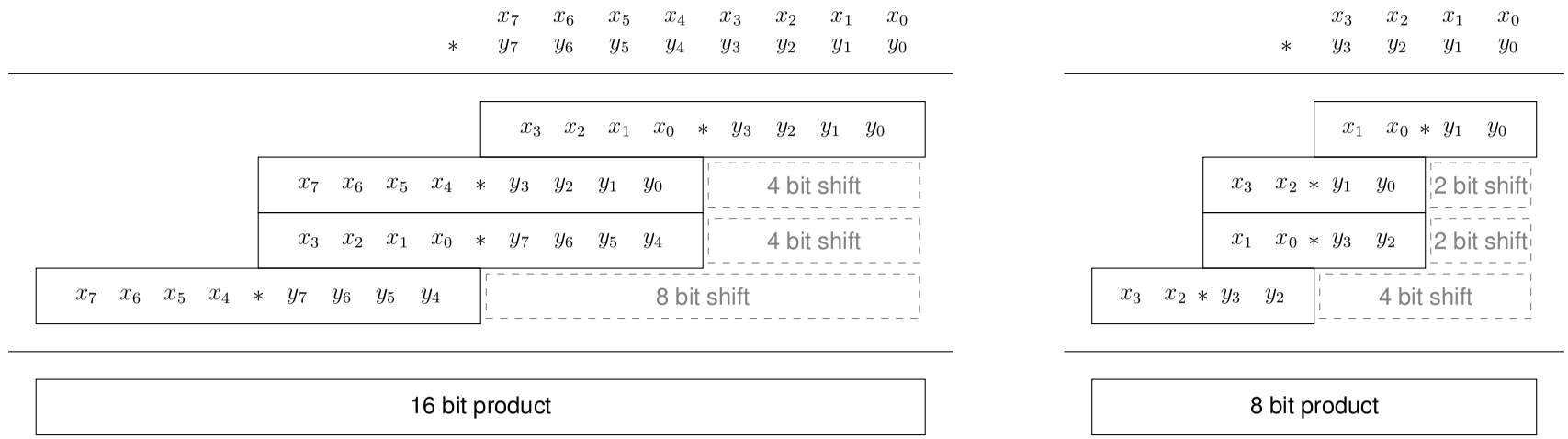


Figure 3.1: Recursive 8-bit multiplier constructed with 4-bit multipliers (left), wherein each 4-bit multiplier constructed with 2-bit multipliers (right) [7]

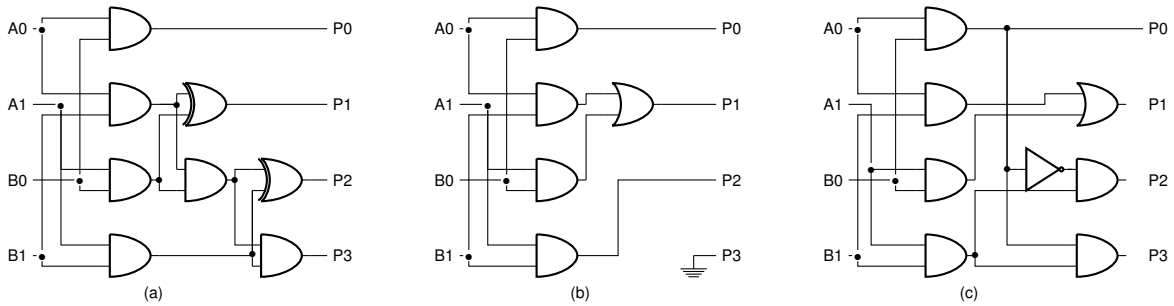


Figure 3.2: Logic diagrams of 2bit multipliers: (a) Accurate 2bit multiplier, (b) Approximate multiplier APXM1 where 3x3 maps to 7 and (c) APXM2, the absolute mirror of APXM1, where 3x3 maps to 11

a \ b	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	7

(a)

a \ b	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	11

(b)

Table 3.1: Result tables for approximate multipliers: (a) APXM1 where 3x3=7 (b) APXM2 with 3x3=11

best tradeoff between area reduced and mean error introduced. The goal is to find two 8-bit multipliers, where one is approximated using the 2-bit multiplier from figure 3.2b and the other uses the 2-bit multiplier from figure 3.2c. With this method, the pairs of absolute mirror multipliers are derived.

In order to find the designs with the best area to mean error tradeoff for the larger 8-bit recursive multipliers of figure 3.1, this thesis performs an exhaustive search on the design space of the unsigned recursive 8-bit multipliers.

The error behaviour of these 8-bit multipliers is simulated by calculating the sum of the error of each individual 2-bit multiplier as shown in equation 3.1.

Design	**Area (μm^2)**
Accurate	9.64
APXM1	7.06
APXM2	8.46

Table 3.2: Area comparison for approximate 2-bit multipliers

ϵ_{2-bit} = magnitude of the error of an individual 2-bit multiplier. If the multiplier is accurate, ϵ_{2-bit}=0, otherwise, ϵ_{2-bit} = +/-2

M_{2-bit} = magnitude of this 2-bit multiplier

P = probability that the error case occurs

$$Error_{2-bit} = \epsilon_{2-bit} * M_{2-bit} * P \tag{3.1}$$

$$Error_{8-bit} = \sum_{15}^{i=0} Error_{2-bit_i} \tag{3.2}$$

For each 2-bit multiplier the error case occurs when both inputs are 3_{10}. The magnitude is determined by the column in the 8-bit multiplier where the approximated design is inserted, as the expected error will be larger if bits with a higher significance are approximated.

The probability of each input occurring is determined by the distribution of the inputs for the multiplier. The designs with the best tradeoff are determined for both a uniform and a normal distribution of input values.

For the area characteristic, the difference in area between an accurate and an approximate 2-bit multiplier is subtracted from the total area of an accurate design, for each instance of an approximate multiplier appearing in the design. The area of each of the 2-bit multiplier designs is given in table 3.2. The area of the addition tree of the multiplier is ignored for this comparison, since it is not affected by a change in the type of 2-bit multiplier.

These error and area characteristics are determined for each combination of accurate and approximate 2-bit multipliers in the overall 8-bit design. For example, lets examine the case where only the top right 2-bit multiplier in the overall 8bit design is approximated with the multiplier from figure 3.2b, and the distribution of input values is uniform. The maximum error $\epsilon = -2$, the probability of this error case occuring $P = \frac{1}{16}$ and the magnitude $M = 1$, giving an expected error $Error_{2-bit}$ of this 2-bit multiplier of $Error_{2-bit} = \frac{1}{8}$. Since the rest of the 2-bit multipliers in this example are accurate, the $Error_{8-bit}(y)$ of the overall multiplier is also $\frac{1}{8}$. The area x of this multiplier is equal to $x = area_{accurate} - 1 * area_{difference} = 16 * 9.64 - 1 * (9.64 - 7.06) = 151.66$. These points x and y are plotted in the graph in figure 3.3 (green dot), together with the value pairs of the complete design space exploration for this method. Each of the points represents the area and mean error of a design with a certain subset of the sixteen 2-bit multipliers approximated. From this graph, sixteen designs are shown to have a best tradeoff between area and error for this approximation strategy.

Sixteen designs appear, because for every design that is analyzed, anywhere between one and sixteen of the 2-bit multipliers are approximated. The upper leftmost

dot on the red line indicates the only, and therefore automatically the best, design that has all sixteen smaller multipliers approximated. Therefore, the area is the lowest, but the error introduced is the largest of all designs. The rightmost column of blue marks indicate all designs where just one 2-bit multiplier is approximated. All of the designs in this column have an equal area. Their respective mean error is different however, because the error depends on the magnitude of the multiplier that was approximated. The single design with the lowest mean error in this column is of course the design with the best tradeoff in this column.

One feature that stands out in figure 3.4 is the appearance of two point clouds in both graphs. This clear distinction between the upper and lower point clouds is caused by the approximation of the most significant 2-bit multiplier. Using the formula from equation 3.1 and knowing that the magnitude for the most significant 2-bit multiplier equals 4096, the contribution to the error by the most significant 2-bit multiplier is 512 ($\epsilon_{2-bit} = 2, P = \frac{1}{16}$).

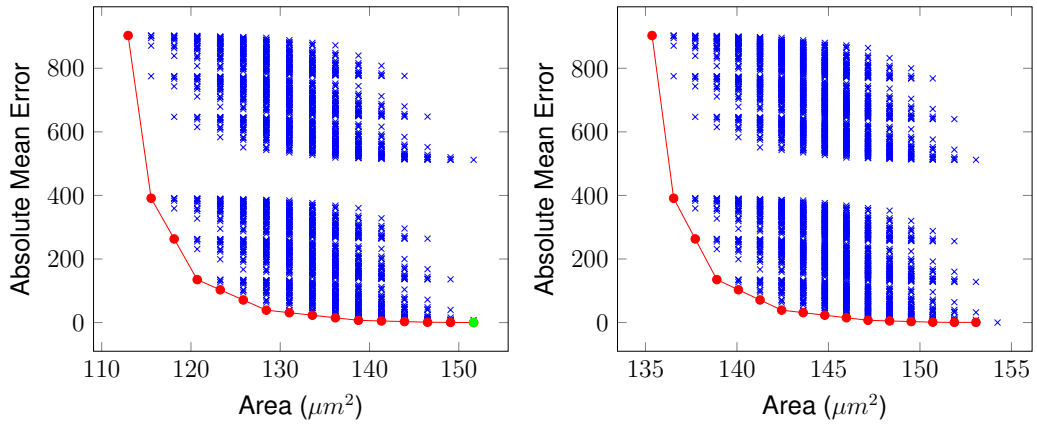


Figure 3.3: Design space overview of approximate 8bit recursive multiplier: (a) Using design APXM1 from 3.2b. (b) Using design APXM2 from 3.2c. Pareto optimal designs are indicated with the red line. The green dot is the example calculation from the text.

The same exploration is performed with the use of multiplier APXM2. This again leads to sixteen multipliers with a best area versus mean error tradeoff. The sixteen best designs from both APXM1 and APXM2 analysis are shown in figure 3.4. Each combination of a red and a blue dot in this graph that have equal absolute mean error, can be combined in an absolute mirror self healing MAC. This leads to a total of sixteen MAC designs from this design strategy.

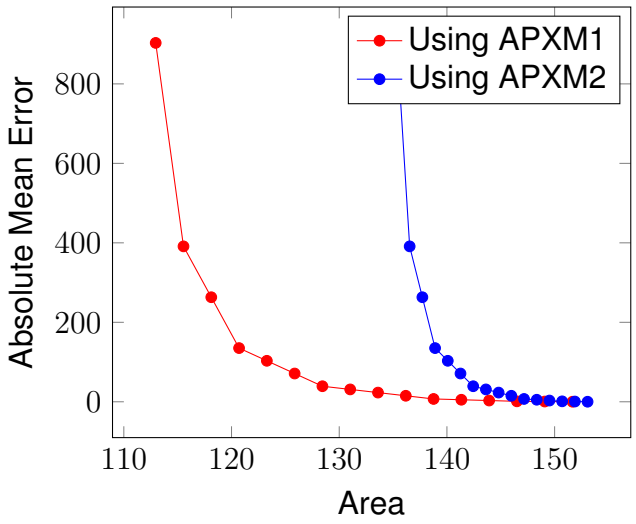


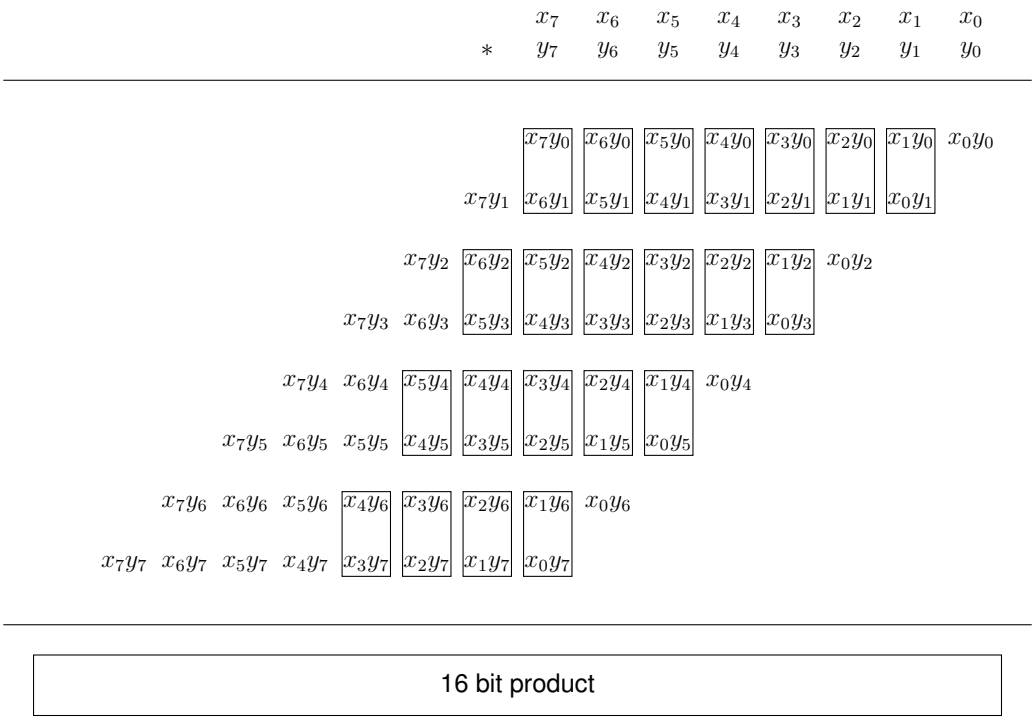
Figure 3.4: Pareto optimal designs for 8bit recursive multipliers with uniform input distribution

3.2 Mean error mirror multipliers

The previous process described a method of determining best designs for a given approximation strategy. This resulted in the selection of multipliers for use in an absolute mirror MAC. However, various multiplier designs have already been proposed that have a better area error tradeoff than the recursive 8-bit designs [14]. The downside to these multiplier designs is that it is not straightforward to design an absolute mirror for these multipliers. If they can even be constructed, their area can be much larger, sometimes even larger than the area of an accurate multiplier. One such multiplier is the proposed design from [14], of which the design is depicted in figure 3.5. In this multiplier the additions of certain pairs of partial products are approximated by utilizing OR gates instead of accurate half adders, so it will be referred to as the OR gate multiplier.

The comparison of the mean error and area characteristics of this multiplier to the earlier derived features of the recursive multipliers is shown in figure 3.6. The absolute mirror of the OR gate multiplier, which is designed as part of this thesis, is depicted too. This time, the area of addition tree for the recursive multipliers is included. It is clear that the OR gate multiplier (green dot) performs better than the recursive multipliers, but its absolute mirror (black dot) does not.

Developing an absolute mirror multiplier for this OR gate multiplier design is possible, but not directly straightforward. The approximation in this design is introduced in the addition tree, by approximating the half adder with logic OR gates. Compared to an accurate half adder which consists of an XOR gate and an AND gate this approach using an OR gate for the approximation is clearly saving area. As for the



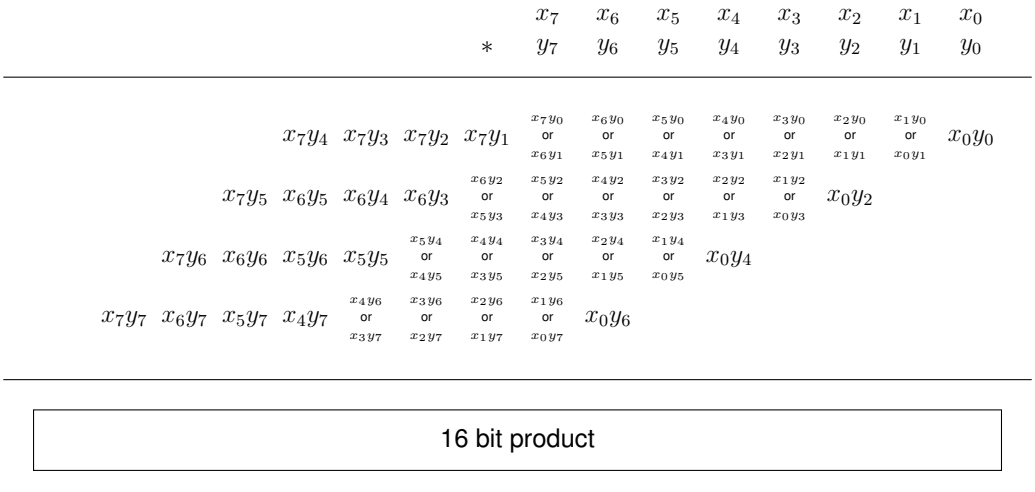


Figure 3.5: From an accurate multiplier (top) to an approximate multiplier (bottom) using logic OR-gates instead of half-adders in the blocked spaces

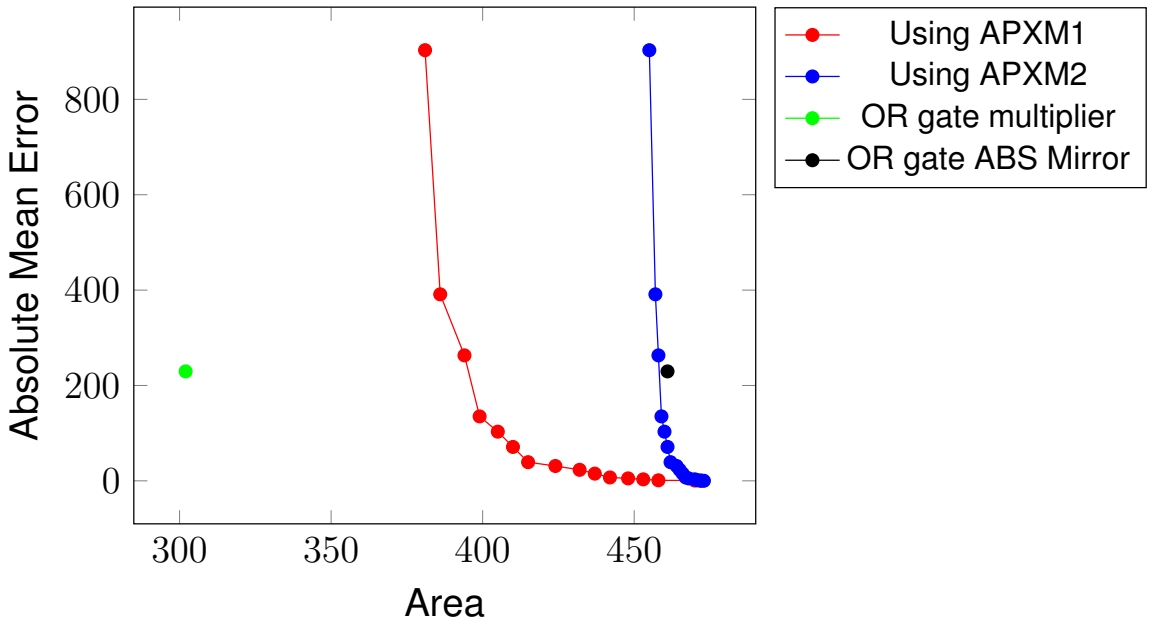


Figure 3.6: Pareto optimal designs for 8-bit recursive multipliers(blue and red), compared with the OR gate multiplier (green) and the absolute mirror of the OR gate multiplier (black)

error behaviour, the truth tables for both designs are shown in table 3.3. From this table it is clear that there exists one error case being when both inputs are 1. In this case the OR gate approximation makes and error of magnitude -1. The gate level designs of all the half adders in the table are shown in figure 3.7.

a	b	a+b	a OR b	abs mirror OR
0	0	00	00	00
0	1	01	01	01
1	0	01	01	01
1	1	10	01	11

Table 3.3: Truth tables for half adder function and or gate approximation for inputs a and b

In order to create an absolute mirror multiplier we need to invert the error introduced by this approximation exactly. The error made by the OR-gate approximation is in one case, and always on the most significant bit of the two bit outcome. In the resulting absolute mirror design, it is desirable that the mirroring behaviour is absolute, independent on the input distribution. Therefore, the mirror design of this OR gate-adder must make an equal, but opposite error on the same input case of both inputs a and b being 1. Also the magnitude of the error must be equal. The design of this approximate half adder (HA) will therefore be equal to an accurate HA, except

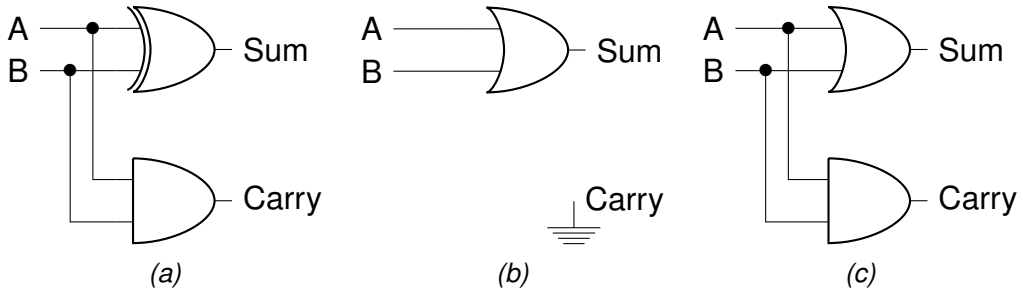


Figure 3.7: 2bit Half Adders(HA): (a) Accurate HA (b) Approximate HA using an OR gate [14] (c) Absolute mirror HA of (b) introduced in this thesis

when both inputs are 1, where the absolute mirror HA will return 11 instead of 10. The resulting absolute mirror HA thus uses an OR gate and an AND gate, as shown in 3.7c. Implementing these absolute mirror HAs into the OR gate multiplier yields the multiplier that corresponds with the black dot in the graph 3.6. This is clearly worse that the recursive designs and another approach is needed if we want to use the OR gate multiplier for a self healing MAC.

The observation can be made that a self healing MAC design can still work with a self healing approach, even when the multipliers are not absolute mirrors. If the number of inputs is large enough, the mean error of the MAC should still approach zero, if the pair of multipliers used in the self healing MAC have an inverse mean error. The OR gate multiplier has already been used in an approximate MAC design in [9]. However, some static error correction value was added and the multipliers partial product matrix was truncated. Moreover, it was not a parallel MAC, but a serial one.

3.2.1 Mean error mirror MAC

If all these techniques are combined, a design with a better tradeoff compared to the discussed absolute mirror MAC may be found. As a proof of concept, a design example is proposed that combines the OR gate multiplier from figure 3.5 with the techniques of self healing, truncation and static error correction. The multiplier it is paired with, in order to create a self healing MAC design, will be an input truncated multiplier with the four least significant bits truncated from the inputs. The advantage of this multiplier is its small size, but it also generates a large error. To compensate for this error, static error correction is applied to this multiplier. The static error correction value (SECV) added to the result of this multiplier is chosen specifically, such that the resulting mean error of the input truncated multiplier after static error correction is exactly the inverse of the mean error of the OR gate multiplier(equation 3.3). The exact value SECV also depends on the input distribution, since the mean

error of both multipliers in this design changes depending on this distribution. These two multipliers are combined in a self healing MAC, as shown in figure 3.8. In this figure, the OR gate multiplier is M1, and the input truncated multiplier is M2.

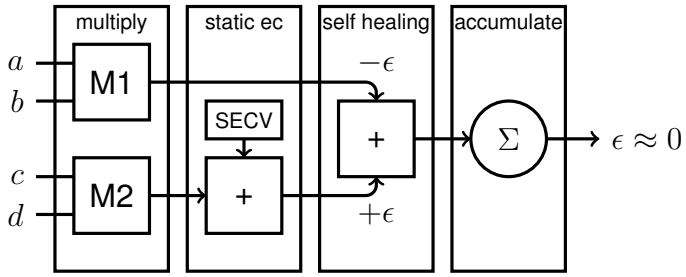


Figure 3.8: Self healing MAC structure with multipliers M1 and M2. M1 is implemented with the OR gate multiplier from 3.5, with mean error $-\epsilon$. M2 is implemented as an input truncated multiplier like Figure 2.2c. Static error correction is applied to the result from M2 to get the desired error $+\epsilon$ for the self healing stage.

ϵ_{M1} = mean error of Multiplier M1
ϵ_{M2} = mean error of Multiplier M2

$$SECV = abs(\epsilon_{M1}) + abs(\epsilon_{M2}) \tag{3.3}$$

Chapter 4

Proof of concept

This chapter aims to show the feasibility of the mean error mirror design approach for MAC as proposed at the end of the previous chapter. To support this design approach, a proof of concept (POC) mean error mirror design is compared with two existing design approaches. These two approaches are the absolute mirror MAC (two designs) and the static error correction MAC (one design). The POC combines multiple techniques for approximation and error correction. One specific design is chosen to show the effectiveness of the mean error mirror approach.

4.1 Experimental setup and tool flow

Figure 4.1 shows the experimental setup to study the quality-efficiency trade-off. Quality analysis has been performed by implementing behavioural models of the proposed designs in Matlab. Accuracy results are generated by calculating a multiply accumulate result for specific input vector lengths repeatedly.

The Synopsys Design Compiler has been used to assess the area costs for the TSMC 40nm Low Power technology library. For verification of the functionality of the designs and generation of SDF files, Questasim has been used in a combination with Matlab models of all proposed MAC designs.

4.1.1 Considered MAC designs

The MAC designs that are evaluated are constructed for the following four categories, based on the results from the multiplier analysis in the previous chapter. The basic architecture of each category is shown in Figure 4.7. A more detailed overview of these MAC designs is given in Appendix A.

Firstly for the absolute mirror multiplier (AMM) strategy, the sixteen pareto optimal multiplier pairs from figure 3.4 are combined into sixteen self healing MACs. The

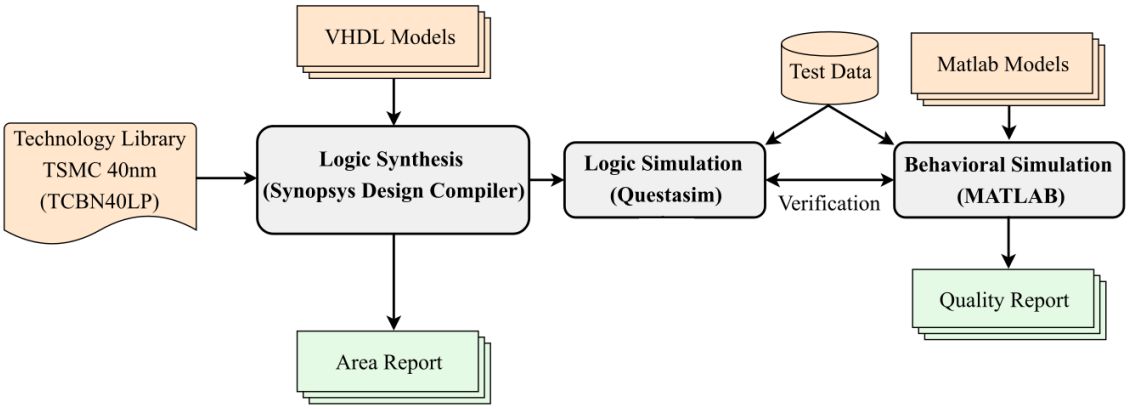


Figure 4.1: Experimental setup for area and error analysis

number given to the design indicates the number of smaller 2bit multipliers that is approximated each of both the 8bit multipliers. These designs are referred to as the MAC_REC_x designs.

Secondly, six designs are evaluated which apply the static error correction method, referred to as MAC_SEC_x designs. These six designs all use the OR gate multiplier from Figure 3.5. Each design has zero to five bits truncated from the PPM of both multipliers. The truncation level is indicated by the number in the name of the design. Two multipliers of the same truncation level are paired to reach the design of a parallel MAC, and this MAC will implement static error correction on the accumulator. The static error correction value is adjusted for vector length and truncation level.

The third design that is shown utilizes the OR gate multiplier and its absolute mirror, as described in the previous chapter. This design has the name MAC_OR_ABS.

Finally the mean error mirror designs as proposed in this thesis are chosen as a proof of concept. These designs have the MAC_MEM_x as a reference name, where the number in the design name again represents the truncation level. These self healing MACs are implemented with the OR gate multiplier and the input truncated multiplier. The truncation level only applies to the OR gate multplier. The static error correction value is adjusted for each combination of input distribution and truncation level.

4.2 Quality analysis

To evaluate the error behaviour of these circuits, a Matlab simulation is performed on all designs. In these simulations every design is provided with input vectors of

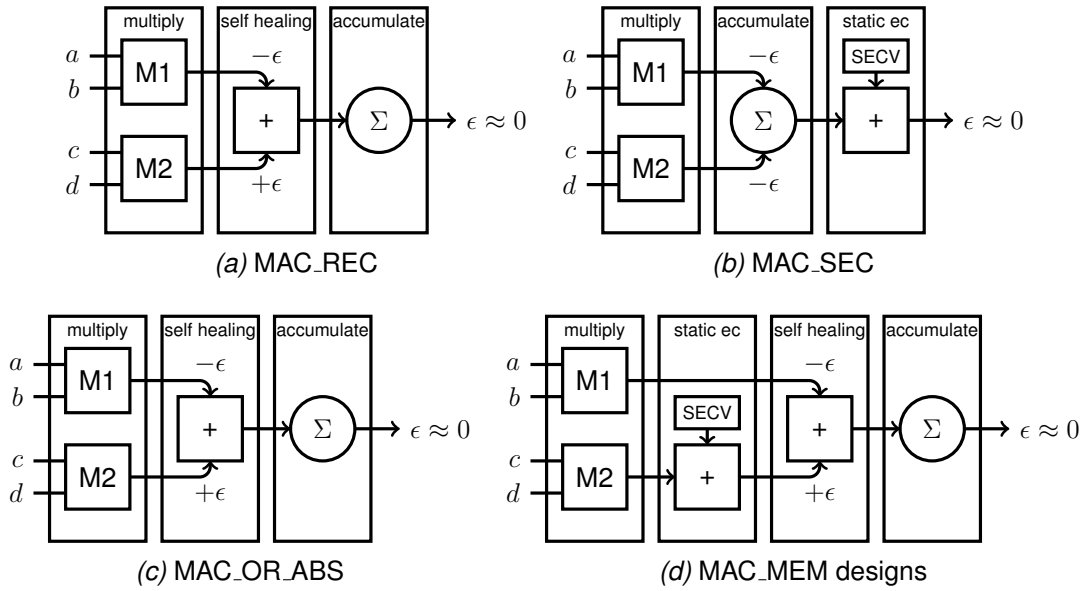


Figure 4.2: Considered parallel MAC designs: (a) and (c) use the same absolute mirror strategy, but (a) utilizes the recursive multipliers from 3.6 and (c) utilizes the OR gate multiplier and its absolute mirror. (b) Applies static error correction and (d) implements the proposed mean error mirror strategy

various sizes. For each vector size, each datapoint in the result graphs present an average over a thousand complete MAC operations. The vector sizes range from 2^0 to 2^{10}, where a vector size of 2^x means that each of the four inputs of the parallel MAC designs receives 2^x inputs. The results of these simulations are analyzed to determine the Mean Error (ME) of each design, as well as the Root Mean Square (RMS) of the error of each design. Both the ME and the RMS values are normalized for their input vector lengths, meaning these error metrics are divided by the length of the input vector. The ME value is chosen to indicate the expected size of the error, which is a measure for the accuracy of the design. The RMS values is chosen to show the spread of the error around the mean, as a way to indicate precision.

y_i = approximated value of MAC operation i

x_i = accurate value of MAC operation i

n = number of MAC operations, in this case always 1000.

$$ME = \frac{\sum_n^{i=1} y_i - x_i}{n}$$

$$RMS = \sqrt{\frac{\sum_n^{i=1} (y_i - x_i)^2}{n}}$$

Both a normal and a uniform distribution of input values are considered. The results are shown in graphs that are split by design strategy for readability.

The graphs in figures 4.3 and 4.4 show the development of the normalized ME and RMS over increasing vector lengths. The values in these input vectors are normally distributed. The graphs in figures 4.5 and 4.6 show the same metrics, but this time for a uniformly distributed input.

The graphs 4.3 and 4.5 that represent the mean error, show that the mean error stabilizes if the vector lenghts are increasing. Also, the more aggressive approximating designs have a higher mean error, especially when the input vector is relatively small.

The figures 4.4 and 4.6 that depict the development of the RMS value, all show a decrease that corresponds nicely to the increase in vector length. The MAC_SEC and MAC_OR_ABS designs show very similar values, whereas the MAC_MEM designs are less predictable, especially for smaller input vector sizes. This is caused by both multipliers in the MEM design, which generate a relatively large error for low input values. With small input vectors, the probility that a combination of high and low inputs occurs together is smaller, thus the spread of the error will be larger. For the MAC_REC designs, this strongly depends on which design is picked. The three designs with the highest number of approximated 2-bit multipliers start off much worse for smaller vector sizes when compared to the MAC_SEC and MAR_OR_ABS designs. The designs with very few approximated multipliers show a much better error behaviour with respect to the RMS value.

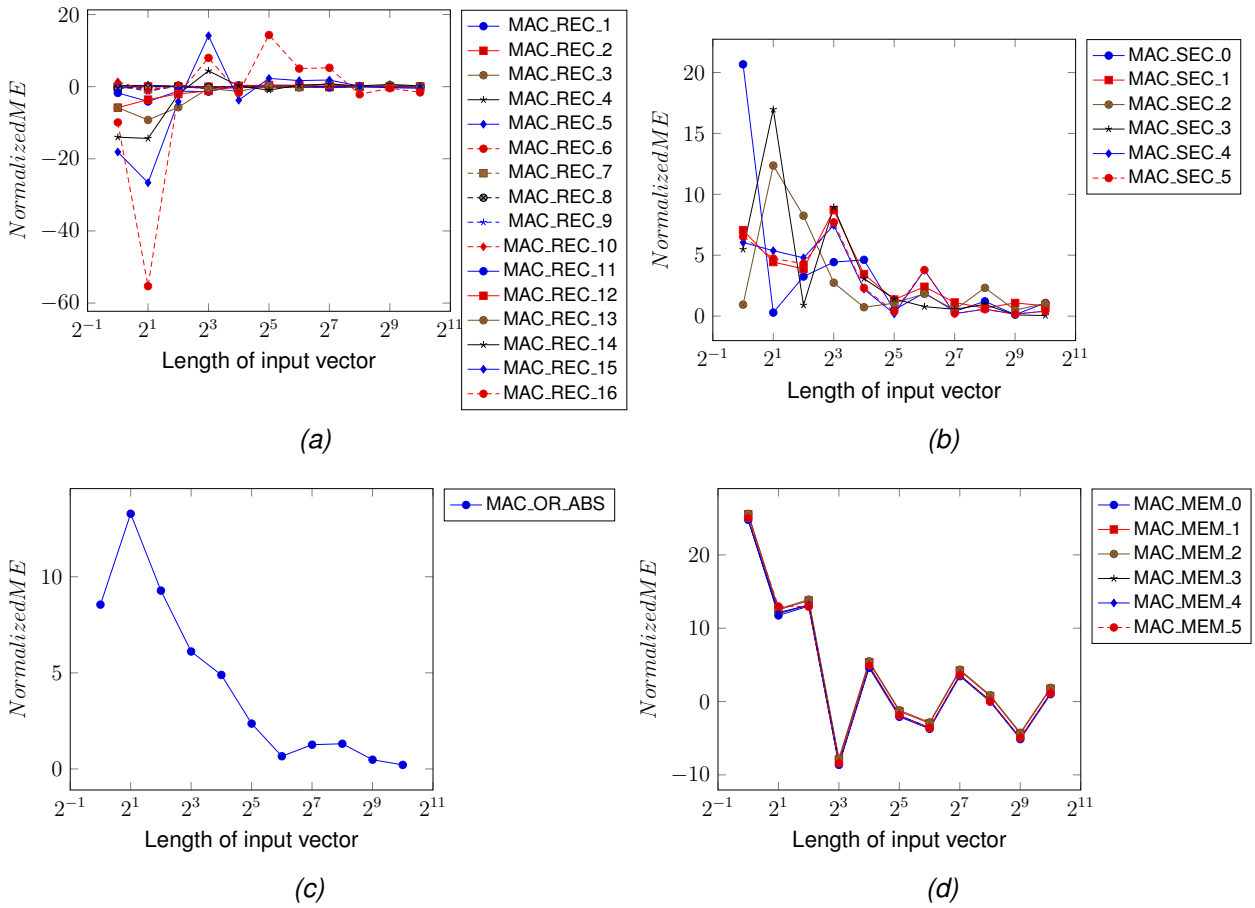


Figure 4.3: Development of Normalized Mean Error (ME) for normal distribution of inputs for the (a) Recursive MACs (b) Static error correction MACs (c) Absolute Mirror MAC (d) Mean Error Mirror MAC

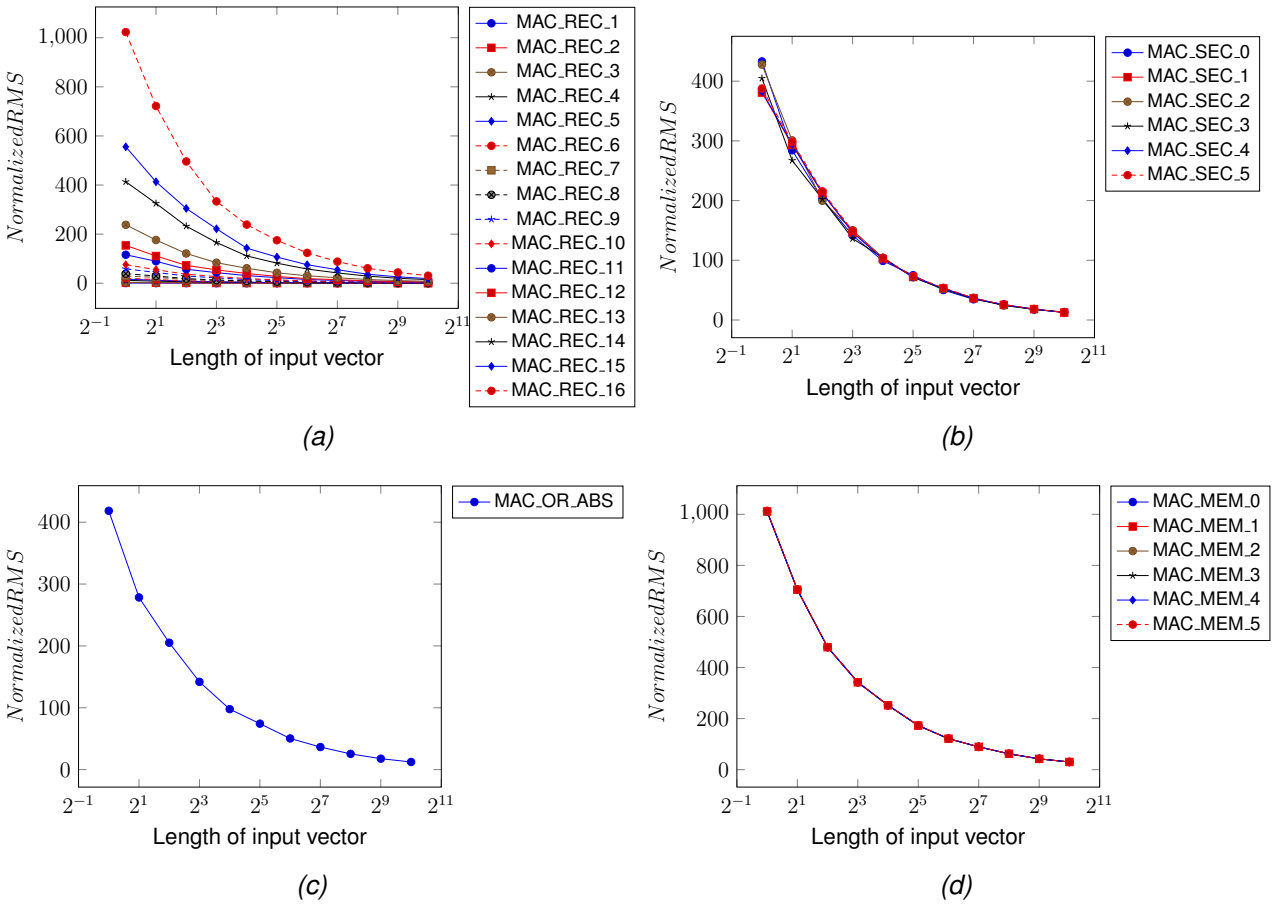


Figure 4.4: Development of Normalized Root Mean Square (RMS) for normal distribution of inputs for the (a) Recursive MACs (b) Static error correction MACs (c) Absolute Mirror MAC (d) Mean Error Mirror MAC

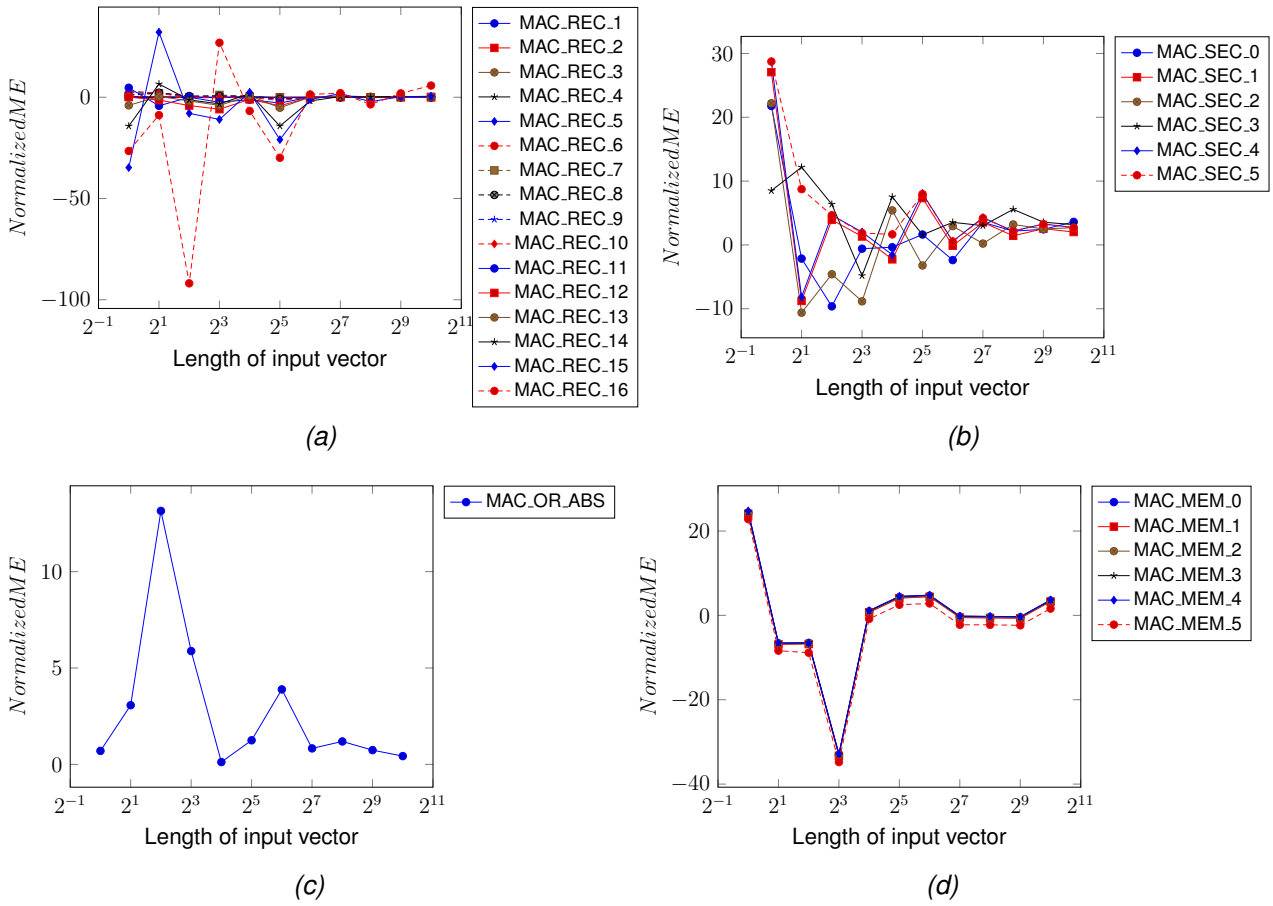


Figure 4.5: Development of Normalized Mean Error (ME) for uniform distribution of inputs for the (a) Recursive MACs (b) Static error correction MACs (c) Absolute Mirror MAC (d) Mean Error Mirror MAC

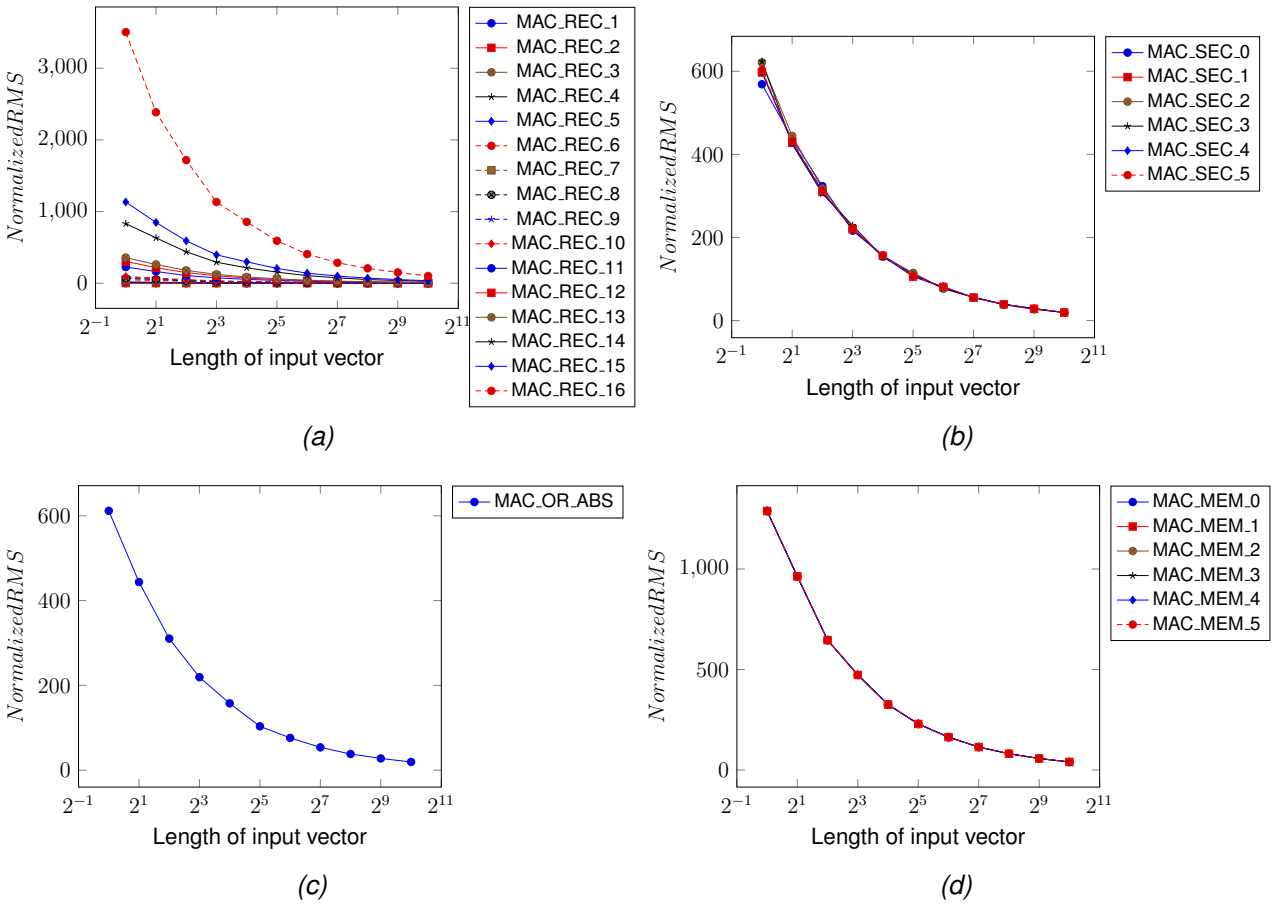


Figure 4.6: Development of Normalized Root Mean Square (RMS) for uniform distribution of inputs for the (a) Recursive MACs (b) Static error correction MACs (c) Absolute Mirror MAC (d) Mean Error Mirror MAC

4.3 Area vs. Error analysis

Using the Synopsys design compiler, the area of each design is determined using a clock speed of 50MHz. These area characteristics are combined with the quality analysis to show their comparative results. These results are shown in the graph in Figure 4.8, where the x axis shows the area of each design, and the y axis shows the mean error (ME). In Figure 4.9 the area of the design is shown on the x axis with the y axis representing the RMS of the designs.

Figure 4.10 again shows the ME against the area, but this time the inputs are normally distributed. Figure 4.11 shows the corresponding graph for the RMS value of the error. The designs shown in these figures are MAC_REC (4.7a), MAC_SEC (4.7b), MAC_OR_ABS (4.7c) and MAC_MEM (4.7d).

Considering the information in the four graphs that represent the area vs. the error metrics, it is clear that the MEM strategy generates the smallest designs, but at the cost of a worse error precision, represented by the RMS value. Still, at least one of the designs can be considered a new contribution, as none of the other design strategies has a better error performance for this small area. Both the MAC_SEC and MAC_MEM designs have a very good error behaviour, since the static error correction has a great effect on reducing the mean error of these designs. The absolute mirror designs of both the MAC_REC and MAC_OR_ABS strategies have a similar low error behaviour, but at the cost of extra area required for these designs.

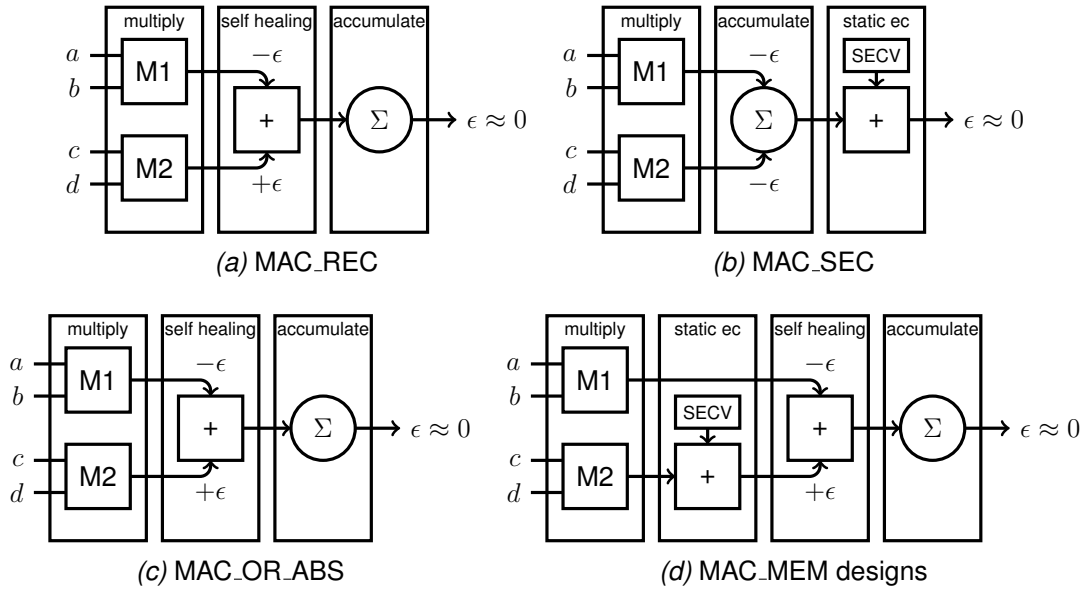


Figure 4.7: Considered parallel MAC designs: (a) and (c) use the same absolute mirror strategy, but (a) utilizes the recursive multipliers from 3.6 and (c) utilizes the OR gate multiplier and its absolute mirror. (b) Applies static error correction and (d) implements the proposed mean error mirror strategy

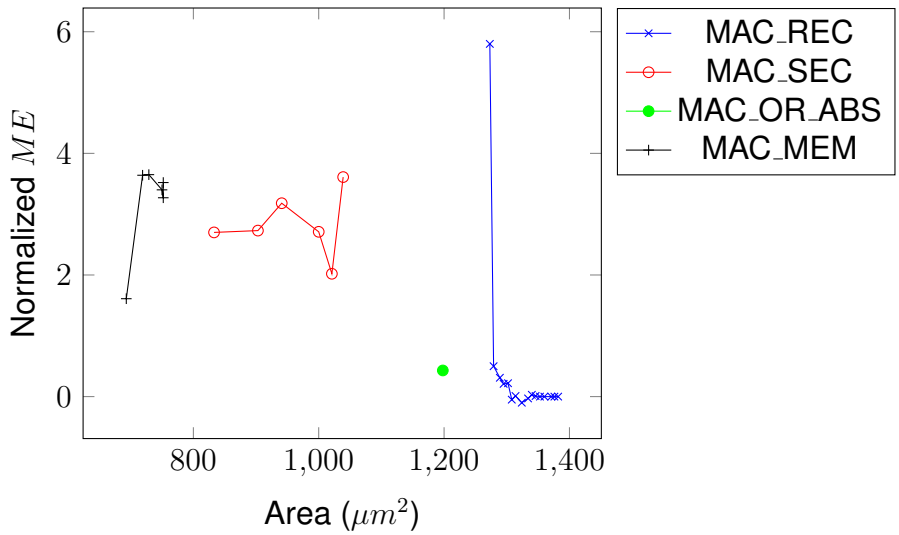


Figure 4.8: Mean Error vs. Area for MAC designs under uniform input for input vector length 2^{10}

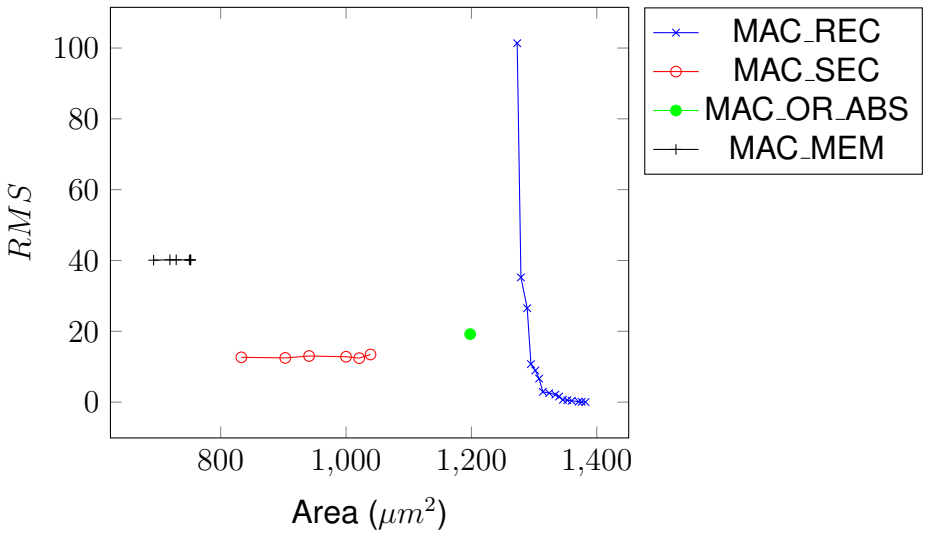


Figure 4.9: RMS vs. Area for MAC designs under uniform input for input vector lenght 2^{10}

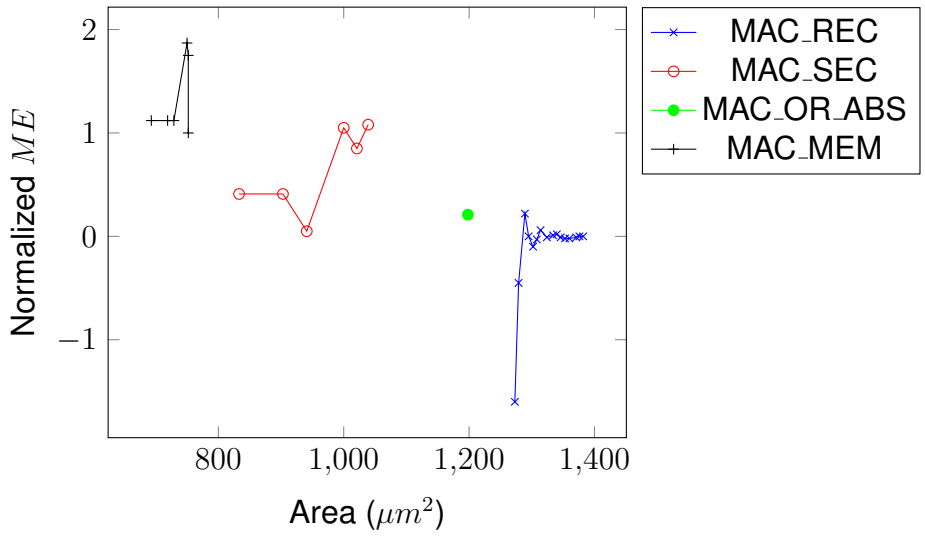


Figure 4.10: Mean Error vs. Area for MAC designs under normal input for input vector length 2^{10}

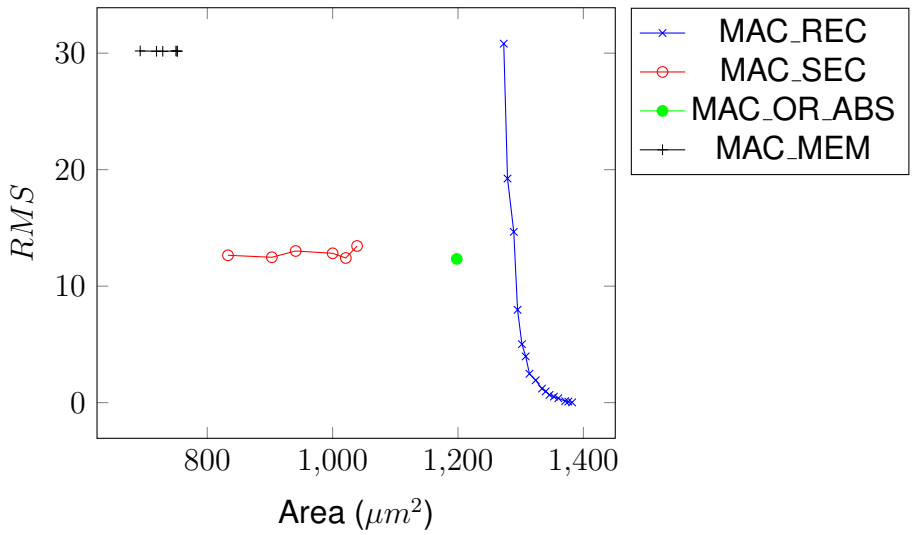


Figure 4.11: RMS vs. Area for MAC designs under normal input for input vector lenght 2^{10}

Chapter 5

Conclusions and recommendations

The main question this thesis has tried to answer, was when considering parallel MAC structures, whether a self healing design utilizing the mean error mirror principle can perform better than similar designs using strategies already shown in literature. The already proven quality of the recursive multiplier design has been tested against three approaches of constructing a self healing MAC with the OR gate multiplier. First, the absolute mirror strategy for self healing was used, secondly a design with static error correction was implemented and finally a mean error mirror (MEM) design was built. The OR gate multiplier was used, as it has a better quality-area tradeoff than the recursive multiplier, thus making it a good candidate for implementing in a self healing MAC.

The mean error mirror MAC as proposed in this thesis combined the OR gate multiplier and an input truncated multiplier. The input truncated multipier was designed with static error correction. The error correction value was set to compensate for both the error introduced by the truncated multiplier, as well as for the error of the MAC structure as a whole.

The proposed MEM design approach yielded one design with a better quality-area tradeoff than existing strategies. Concluding, a new pareto optimal design has been added to the field of existing parallel approximate MAC design strategies.

5.1 Future work

The approach chosen for constructing a mean error mirror MAC was ambitious, but the idea of utilizing two inverse mean error multipliers together which are not an absolute mirror of one another is promising for further investigation. This thesis presents a proof of concept of this design strategy, so the next step is to devise a method for a design space exploration of mean error mirror MACs. Using this method should result in a more complete insight in the viability of the mean error mirror design approach.

A limitation of this research was that the modelling has only been performed on uniform and normal distributions of unsigned inputs. Expanding to signed inputs and including more distributions in the determination of the quality of the designs, might bring some new insights on how to approach this strategy in the future.

Another extension to this research could be the inclusion of power simulations. Although the area results give an indication of the power consumption of the IC designs, performing these power simulations would certainly increase the value of the results presented in this thesis.

Bibliography

[1] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 120.

[2] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[3] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2015.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited - Cross-layer approximate computing: from logic to architectures," in *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16*, 2016.

[5] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2011.

[6] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in *Proceedings of the IEEE Conference on Nanotechnology*, 2013.

[7] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-Space Exploration of Approximate Multipliers."

[8] S.-R. Kuang and J.-P. Wang, "Low-error configurable truncated multipliers for multiply-accumulate applications."

[9] D. Esposito, A. G. Strollo, and M. Alioto, "Low-power approximate MAC unit," in *PRIME 2017 - 13th Conference on PhD Research in Microelectronics and Electronics, Proceedings*, 2017, pp. 81–84.

[10] G. A. Gillani, M. A. Hanif, M. Krone, S. H. Gerez, M. Shafique, and A. B. Kokkeler, "Squash: Approximate square-accumulate with self-healing," *IEEE Access*, vol. 6, pp. 49 112–49 128, 2018.

[11] G. Gillani, “Exploiting error resilience for hardware efficiency – targeting iterative and accumulation based algorithms,” Ph.D. dissertation, University of Twente, 2020.

[12] J. Han and M. Orshansky, “Approximate Computing: An Emerging Paradigm For Energy-Efficient Design.”

[13] S. Mittal, “A survey of architectural techniques for near-threshold computing,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 4, p. 46, 2016.

[14] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, “Energy-efficient approximate multiplier design using bit significance-driven logic compression,” in *Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017*, 2017.

[15] V. Garofalo, N. Petra, D. De Caro, A. G. M. Strollo, and E. Napoli, “Low error truncated multipliers for DSP applications,” in *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008*, 2008.

[16] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014.

Appendix A

Overview of the multiplier designs for approximate MAC

This appendix gives a detailed overview of the approximate multipliers that make up the approximate MAC designs as presented and proposed in this thesis. Of these multipliers their respective mean error is given. With the mean errors of these multipliers, that SECV values of the resulting MACs can be verified. The multipliers that are mentioned are the recursive multiplier, the OR gate multiplier and the input truncated multiplier used in the proposed mean error mirror MAC design.

A.1 Recursive multiplier

The 8-bit multiplier that was analyzed is constructed using sixteen smaller 2-bit multipliers, as shown in figure 3.1. A more simplistic depiction is shown in figure A.1. For each of these 2-bit multipliers, a reference number and their magnitude is shown, which is used in the error calculation as shown in chapter 3.

			7 M = 2^6	4 M = 2^4	2 M = 2^2	1 M = 2^0
		11 M = 2^8	8 M = 2^6	5 M = 2^4	3 M = 2^2	
	14 M = 2^{10}	12 M = 2^8	9 M = 2^6	6 M = 2^4		
16 M = 2^{12}	15 M = 2^{10}	13 M = 2^8	10 M = 2^6			

Figure A.1: Simplified depiction of the 8-bit recursive multiplier. Each block represents a 2-bit multiplier with its reference number and magnitude (M).

	Uniform input		Normal input	
Model name	**Approximated 2-bit multipliers**	**Mean error**	**Approximated 2-bit multipliers**	**Mean error**
mult_rec_apxm1_1	1	-0.125	1	-0.125
mult_rec_apxm1_2	1,2	-0.625	1-2	-0.625
mult_rec_apxm1_3	1-3	-1.125	1-3	-1.125
mult_rec_apxm1_4	1-4	-3.125	1-4	-3.125
mult_rec_apxm1_5	1-5	-5.125	1-4,6	-5.12
mult_rec_apxm1_6	1-6	-7.125	1-6	-7.12
mult_rec_apxm1_7	1-7	-15.125	1-7	-9.197
mult_rec_apxm1_8	1-8	-23.125	1-7,10	-11.274
mult_rec_apxm1_9	1-9	-31.125	1-8,10	-19.264
mult_rec_apxm1_10	1-10	-39.125	1-10	-27.255
mult_rec_apxm1_11	1-11	-71.125	1-11	-35.562
mult_rec_apxm1_12	1-12	-103.125	1-11,13	-43.869
mult_rec_apxm1_13	1-13	-135.125	1-13	-75.792
mult_rec_apxm1_14	1-14	-263.125	1-14	-108.981
mult_rec_apxm1_15	1-15	-391.125	1-15	-142.170
mult_rec_apxm1_16	1-16	-903.125	1-16	-176.675

Table A.1: The approximate recursive multiplier designs and their mean error values. Approximation using APXM1

The overview of the multipliers approximated and the resulting error is given in table A.1. In this table, the 2-bit multiplier that was used for the approximation is the APXM1 from figure 3.2b. The results for approximation with APXM2 gives the same results, but with positive mean error. The multiplier models using APXM2 will be referred to as mult_rec_apxm2_(1-16).

A.2 OR gate multiplier

The basic multiplier that is used for the MAC_SEC, MAC_OR_ABS and MAC_MEM design is the OR gate multiplier from figure 3.5. Truncation is applied to the PPM of this multiplier, like in figure 2.2b. The models and their mean error are shown in table A.2.

Model name	Truncation level	Mean error (uniform)	Mean error (normal)
mult_or_0	0	-229	-137
mult_or_1	1	-229	-138
mult_or_2	2	-230	-139
mult_or_3	3	-233	-141
mult_or_4	4	-240	-148
mult_or_5	5	-256	-166

Table A.2: Approximate OR gate multipliers used in approximate MAC designs.

A.2.1 Absolute mirror of OR gate multiplier

The absolute mirror of the OR gate multiplier was only developed for the mult_or_0 model, as described in chapter 3. As it was developed as an absolute mirror, the mean error are exactly the positive values of the mult_or_0 model. This model is referred to as mult_or_0_abs.

A.3 Input truncated multiplier

Finally, for the proof of concept MAC_MEM designs, an input truncated multiplier is used. The method for generating this multiplier was already shown in figure 2.2c, but for this purpose the four least significant bits were truncated from the input. This multiplier is shown in A.2, and has a mean error of -1856 for both a uniform and a normal input distribution. The name of this multiplier is mult_input_trunc.

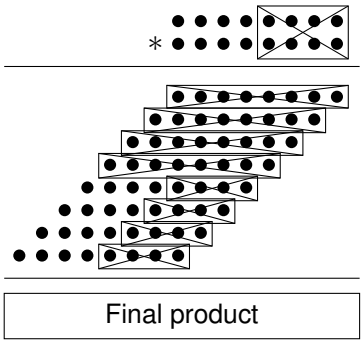


Figure A.2: The input truncated multiplier used in the proposed MAC_MEM designs.

A.4 MAC designs

The table below shows which multipliers make up the various MAC designs of which the area and error characteristics where analyzed in chapter 4. If no SECV is given in the table, no static error correction was applied to this design. The designs and names correspond to the designs in figure 4.7.

MAC design name	**M1**	**M2**	$SECV_{uniform}$	$SECV_{normal}$
MAC_REC_1	mult_rec_apxm1_1	mult_rec_apxm2_1		
MAC_REC_2	mult_rec_apxm1_2	mult_rec_apxm2_2		
MAC_REC_3	mult_rec_apxm1_3	mult_rec_apxm2_3		
MAC_REC_4	mult_rec_apxm1_4	mult_rec_apxm2_4		
MAC_REC_5	mult_rec_apxm1_5	mult_rec_apxm2_5		
MAC_REC_6	mult_rec_apxm1_6	mult_rec_apxm2_6		
MAC_REC_7	mult_rec_apxm1_7	mult_rec_apxm2_7		
MAC_REC_8	mult_rec_apxm1_8	mult_rec_apxm2_8		
MAC_REC_9	mult_rec_apxm1_9	mult_rec_apxm2_9		
MAC_REC_10	mult_rec_apxm1_10	mult_rec_apxm2_10		
MAC_REC_11	mult_rec_apxm1_11	mult_rec_apxm2_11		
MAC_REC_12	mult_rec_apxm1_12	mult_rec_apxm2_12		
MAC_REC_13	mult_rec_apxm1_13	mult_rec_apxm2_13		
MAC_REC_14	mult_rec_apxm1_14	mult_rec_apxm2_14		
MAC_REC_15	mult_rec_apxm1_15	mult_rec_apxm2_15		
MAC_REC_16	mult_rec_apxm1_16	mult_rec_apxm2_16		
MAC_SEC_0	mult_or_0	mult_or_0	$+458$	$+274$
MAC_SEC_1	mult_or_1	mult_or_1	$+458$	$+276$
MAC_SEC_2	mult_or_2	mult_or_2	$+460$	$+279$
MAC_SEC_3	mult_or_3	mult_or_3	$+460$	$+282$
MAC_SEC_4	mult_or_4	mult_or_4	$+480$	$+296$
MAC_SEC_5	mult_or_5	mult_or_5	$+512$	$+332$
MAC_OR_ABS	mult_or_0	mult_or_0_abs		
MAC_MEM_0	mult_or_0	mult_input_trunc	$+2085$	$+1993$
MAC_MEM_1	mult_or_1	mult_input_trunc	$+2085$	$+1994$
MAC_MEM_2	mult_or_2	mult_input_trunc	$+2086$	$+1995$
MAC_MEM_3	mult_or_3	mult_input_trunc	$+2089$	$+1997$
MAC_MEM_4	mult_or_4	mult_input_trunc	$+2096$	$+2004$
MAC_MEM_5	mult_or_5	mult_input_trunc	$+2112$	$+2022$

Table A.3: Combination of multipliers forming the approximate MAC designs