July 2018

ANALOG SIGNAL PROCESSING SOLUTIONS AND DESIGN OF MEMRISTOR-CMOS ANALOG CO-PROCESSOR FOR ACCELERATION OF HIGH-PERFORMANCE COMPUTING APPLICATIONS

Nihar Athreyas
University of Massachusetts Amherst

ANALOG SIGNAL PROCESSING SOLUTIONS AND DESIGN OF MEMRISTOR-CMOS ANALOG CO-PROCESSOR FOR ACCELERATION OF HIGH-PERFORMANCE COMPUTING APPLICATIONS

A Dissertation Presented

by

NIHAR ATHREYAS

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2018

ELECTRICAL AND COMPUTER ENGINEERING

ANALOG SIGNAL PROCESSING SOLUTIONS AND DESIGN OF MEMRISTOR-CMOS ANALOG CO-PROCESSOR FOR ACCELERATION OF HIGH-PERFORMANCE COMPUTING APPLICATIONS

A Dissertation Presented

by

NIHAR ATHREYAS

Approved as to style and content by:

_____
Dev Gupta, Co-Chair

_____
J. Joshua Yang, Co-Chair

_____
Patrick Kelly, Member

_____
Blair Perot, Member

_____
Christopher V. Hollot, Department Head
Electrical and Computer Engineering

DEDICATION

To

Mom, Dad, Lakshmi, My family and My Dearest Friends (*too many to mention by name but, they know who they are*) *who have been a constant source of motivation and support, at times when I needed it* (*or not*)

And

To Tom Brady and the Patriots for the never give up attitude, and without whom Sundays would be very boring.

ACKNOWLEDGMENTS

It gives me immense pleasure to thank those who have made this work possible. I would like to start by thanking Dr. Dev Gupta for believing in me and hiring me as his research assistant 5 years ago. He has had a strong influence on my professional life and he has always pointed me in the right direction, every time I needed it. I'm lucky to have him as my advisor and mentor. I would like to thank Dr. Joshua Yang who co-advised me on this thesis. He always offered valuable suggestions and encouragement. I would like to extend my gratitude to Dr. Patrick Kelly who has always been a constant source of motivation and support. I would like to express my gratitude to Dr. Blair Perot for his valuable time, suggestions and advice and for getting me interested in partial differential equations. I want to thank Dr. Qiangfei Xia for all his support and guidance.

I would also like to thank Jai Gupta and Abbie Mathew of Spero Devices for accommodating me and supporting me in every possible way. It's always a pleasure working with you guys.

I would like to thank the University of Massachusetts, Amherst for providing a wonderful environment and resources for the completion of this project.

A special thank you to my family and all my friends for the support they provided me with when the going got tough and everything else that they have done.

ABSTRACT

ANALOG SIGNAL PROCESSING SOLUTIONS AND DESIGN OF MEMRISTOR-CMOS ANALOG CO-PROCESSOR FOR ACCELERATION OF HIGH-PERFORMANCE COMPUTING APPLICATIONS

MAY 2018

NIHAR ATHREYAS

B.E., VISVESVARAYA TECHNOLOGICAL UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor Dev Gupta and Professor Joshua Yang

Emerging applications in the field of machine vision, deep learning and scientific simulation require high computational speed and are run on platforms that are size, weight and power constrained. With the transistor scaling coming to an end, existing digital hardware architectures will not be able to meet these ever-increasing demands. Analog computation with its rich set of primitives and inherent parallel architecture can be faster, more efficient and compact for some of these applications. The major contribution of this work is to show that analog processing can be a viable solution to this problem. This is demonstrated in the three parts of the dissertation.

In the first part of the dissertation, we demonstrate that analog processing can be used to solve the problem of stereo correspondence. Novel modifications to the algorithms are proposed which improves the computational speed and makes them efficiently implementable in analog hardware. The analog domain implementation provides further speedup in computation and has lower power consumption than a digital implementation.

In the second part of the dissertation, a prototype of an analog processor was developed using commercially available off-the-shelf components. The focus was on providing experimental

results that demonstrate functionality and to show that the performance of the prototype for low-level and mid-level image processing tasks is equivalent to a digital implementation. To demonstrate improvement in speed and power consumption, an integrated circuit design of the analog processor was proposed, and it was shown that such an analog processor would be faster than state-of-the-art digital and other analog processors.

In the third part of the dissertation, a memristor-CMOS analog co-processor that can perform floating point vector matrix multiplication (VMM) is proposed. VMM computation underlies some of the major applications. To demonstrate the working of the analog co-processor at a system level, a new tool called PSpice Systems Option is used. It is shown that the analog co-processor has a superior performance when compared to the projected performances of digital and analog processors. Using the new tool, various application simulations for image processing and solution to partial differential equations are performed on the co-processor model.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

From the second half of the twentieth century analog computation has been eclipsed by digital computation. The impact of digital computation has been so great that the use of the word "computing" conjures up certain assumptions and images of digital electronics. This has been so deeply ingrained in our minds that we rarely question it. However, in the recent years this has changed. Moore's law is coming to an end and the efficiency of computation is not increasing very rapidly. Wringing more performance out of a digital system through sequential execution, Von Neumann architectures, multicore processors and near threshold voltage computation is becoming extremely difficult. Other factors contributing to this are the limited availability of on chip power and rise of dark silicon. This raises fundamental questions on the design of computing architectures.

One very good example of where these effects are being felt is the imaging industry. The imaging industry is going through a huge change. The growth is fuelled by applications in consumer, automotive, medical and industrial markets. Also, contributing to the growth are the emerging applications such as machine vision and deep learning in wearables, drones, Internet of Things etc. Another area that is gaining traction is the use of stereo cameras, camera arrays and light-field cameras to perform computational imaging tasks. On the other end of the spectrum are applications in High Performance Computing (HPC) such as scientific simulation. A common thread in the emerging applications being considered is the requirement of real-time processing to be performed in severely constrained environment of Size, Weight and Power (SWaP). Also, these applications involve collecting and processing huge amounts of data which will lead to

communication bottlenecks between the processor and memory. Due to the high computational requirement, these applications are being implemented on specialized digital hardware components. These digital hardware accelerators have high compute capability, but it comes at a price of high power consumption and large size. For example, achieving real-time object detection on a state-of-the-art GPU like Tegra X1 from Nvidia would require a power consumption of 3.5 Watts [70]. This shows that the current imaging architectures and digital image processing solutions will not work in all situations because they will not be able to handle the high computational loads and meet the SWaP requirements simultaneously.

Since the digital processing approach fails to achieve the required results, a promising alternative is to use analog processing. Analog computation has a long and an illustrious history. The south-pointing chariot which was invented in ancient China during the first millennium BC can be considered as the earliest analog computer. The Antikythera mechanism which dates back to *circa* 100 BC was an orrery and devices as complex as this would not reappear till a thousand years later. The slide rule dominated science and engineering calculations for hundreds of years. While analog signal processing generally cannot achieve the same level of accuracy as digital processing, it has the advantages of higher speed, smaller area and lower power consumption. As a result, analog processors can be built that can perform massive parallel processing. This will result in significant speed-up of computation when compared to a digital system.

This dissertation is aimed at demonstrating the fact that analog processing is a viable solution to the above-mentioned problems. The dissertation consists of three major parts. The first part (CHAPTER 2) discusses the implementation of stereo correspondence algorithms in the analog domain and its advantages. The second part of the dissertation (CHAPTER 3) talks about the design and implementation of a prototype of an analog processor using commercial off-the-

shelf components which can be used for low-level and mid-level image processing tasks that are generally used in machine vision algorithms. The third part of the dissertation (CHAPTER 4 and CHAPTER 5) discusses the design of a memristor-CMOS analog co-processor that can handle floating point VMM operations. This co-processor can be used to accelerate High-Performance computing applications.

1.1. Analog signal processing solution for image alignment

Humans have binocular vision, and this enables us to have *stereopsis* or the ability to perceive depth information. A similar technique is used in the field of computer vision and photography where multiple cameras take images of the same scene. Figure 1 shows an image pair from the Middlebury dataset. Both these images are of the same scene but are captured from different camera positions. These images, called stereo images are used for a variety of purposes like calculating the depth of the objects present in the scene, refocusing, simulating the effect of optical zoom etc. Stereo image alignment can also be used for stitching images to create panoramas, for video stabilization, scene summarization etc.

(a) (b)

Figure 1: Stereo Image Pairs from Middlebury Dataset (a) Left Image (b) Right Image

Whatever the application, one of the most important steps in stereo image processing is to find correspondence between the points in the two images which represents the same 3D point in the scene. This is called correspondence problem in image alignment and this has been an active area of research for many years now and a lot of stereo correspondence algorithms have been developed. However, most of these algorithms are very slow. The reason for these algorithms being slow is the very high computational requirement. In this dissertation, we propose analog image processing as a solution to address the high computational load of these stereo correspondence algorithms.

The implementation of two stereo correspondence algorithms are discussed in this work. The first algorithm is a patch based stereo correspondence algorithm and Normalized Cross-correlation (NCC) is chosen as the similarity measure for this approach. NCC is very robust to noise and changes in the image intensity values. The second algorithm is a per-pixel algorithm which produces a finer disparity map than the patch based approach. Sum Absolute Differences (SAD) is used as a similarity measure for this algorithm. Both these algorithms have a high computational intensity. Novel modifications to the algorithms are proposed which improves the computational speed without compromising the performance and making them efficiently implementable in hardware. New circuit architectures that can be used to implement the modified NCC algorithm and modified SAD algorithm in the analog domain are also proposed. The analog domain implementation provides further speedup in computation and has lower power consumption than a digital implementation.

1.2. Analog Signal Processing Solution for Machine Vision Applications

Computer vision was a research technology for a long time, but over the last few years it has gone through a huge change and is being widely deployed. Long-standing and important problems in the field of computer vision like object detection and recognition, depth estimation etc. are becoming mainstream now and are frequently used in machine vision applications. Most of the machine vision algorithms involve performing low-level and mid-level image processing tasks like image enhancement, filtering, segmentation and classification of objects.

The accuracy achieved by analog processing is often sufficient for these image processing tasks. Another common feature in these algorithms is the requirement for parallel processing. For example, most of the low-level and mid-level image processing tasks can be implemented through convolution of pixel intensities with filter kernel values. In these cases, each image pixel can be processed independently and in parallel. Hence, analog processing is a suitable candidate for these tasks. There have been successful attempts to develop analog vision chips that are faster than digital systems using a variety of technologies ranging from CMOS devices, to the more recently developed spintronic based Nano devices. Some of the vision chips described were designed as application specific integrated circuits while others show simulation results. The design of application specific integrated circuits is a long and expensive process. Simulation results are a good indicator of performance but cannot always indicate actual circuit performance due to many constraints, and the results also depend on how well the circuit models have been defined. In this work, we have taken a different approach. We have built a prototype of an analog processing board using commercial off-the-shelf (COTS) components. The advantages of developing such a prototype board include flexibility, cost savings, decreased time-to-market and lower risk than a direct custom integrated circuit design. It is also better than simulating with circuit models alone. It will help us understand analog processing techniques required for machine

vision applications and create better circuit architectures for an integrated circuit design. The board has three analog channels which can be used to perform various operations such as convolutions and correlations in the analog domain. In this part of the dissertation, we focus on providing experimental results that demonstrate the functionality of the analog processing board. We also propose the development of an integrated circuit design called the Analog Space-Time Convolution Processor (ASTCP) based on a similar architecture and compare the performance of the proposed analog hardware accelerator against existing digital accelerators and other analog processors proposed in literature.

1.3. Memristor-CMOS Analog Co-Processor for Acceleration of High-Performance Computing Applications

Emerging applications in HPC such as deep learning, image processing and scientific simulation simultaneously require high computational speed and low power consumption. With the transistor scaling coming to an end and Moore's law slowing down, existing digital hardware accelerators will not be able to meet these requirements. This opens an opportunity for custom hardware with novel device types and disruptive architectures to serve unmet needs. Research has shown that analog computation with its rich set of primitives and inherent parallel architecture can be faster, more efficient and compact for some of these applications. In this work, we propose the development of an analog co-processor to accelerate the solutions to HPC applications by invoking VMM in memristor-CMOS crossbar arrays. VMM computation underlies major HPC applications such as solving partial differential equations and performing deep learning inference and training. In this work, we develop an analog co-processor architecture that can handle floating point computation. The architecture is very modular and scalable. The analog co-processor model proposed in this work uses a TaO_x memristor and a 0.25µm TSMC model of a transistor. A TaO_x model based on experimental data of electronic transport and large switching

dynamics was first developed by [96]. Here, we extend the TaO_x device modeling to develop a circuit model in PSpice. The PSpice model was used to simulate the electrical behavior of the memristor crossbar array. However, the PSpice models alone cannot support complex application simulations. Mathematical models for the memristor crossbar array can be developed in MATLAB and can be used for some application simulations. However, the MATLAB models are considered to be too ideal even with non-idealities included in them. To circumvent this issue, we use a new EDA tool called PSpice Systems Option [17] developed by Cadence and Mathworks. It performs integrated co-simulation of MATLAB, Simulink and PSpice. It allows for mixed analog and digital simulation, and for computations within Simulink models to be mapped into PSpice. Using the new tool, we are able to perform application simulations at a circuit level which adds credibility to the design of the analog co-processor.

CHAPTER 2

ANALOG SIGNAL PROCESSING FOR IMAGE ALIGNMENT

2.1. Patch Based Stereo Correspondence Algorithm

All stereo correspondence algorithms can be broadly classified into intensity-based algorithms and feature-based algorithms. In feature-based algorithms, features such as edges and contours are extracted from both stereo images and then a correspondence is established between them. In intensity-based algorithms, blocks of pixels from one image are compared to blocks of pixels from other images and a similarity measure such as correlation or sum absolute difference is used to find the best matching block. The disparity value corresponding to the best matching block or patch is assigned to all the pixels of that patch. Hence, this is known as a patch based stereo correspondence algorithm. Each of these algorithms have their own advantages and disadvantages. Both algorithms are used widely.

The Intensity based algorithms are very simple to implement, they are robust, and they produce dense depth maps. They fail to perform well when the distance between the stereo cameras is too large or if there are rotations and shears in the stereo images. However, the major drawback of the intensity-based algorithms is that they are highly computational and hence it will be the algorithm of interest. In this dissertation, we address the issue of high computational load of the intensity-based algorithms through novel modifications to the algorithm and by the way of analog signal processing.

2.2. Review of Related Work

There has been a lot of research done on stereo image registration techniques as it relates to multiple fields like computer vision, medical imaging, photography etc. Image registration is the process of aligning two or more images of the same scene by applying geometric transformations. A variety of algorithms, both feature-based and intensity-based have been developed. In [16], the author provides a survey of different image registration techniques used in various fields.

In this study, we are mainly concerned with the implementation of an intensity-based image alignment algorithm in hardware. There has been some work done in this regard, but most of them are improvements to the old algorithms and some are digital hardware implementations of these algorithms. In [61] Lewis proposes a fast, normalized cross-correlation algorithm, which reduces the computational complexity of the normalized cross-correlation algorithm through the use of sum table methods to pre-compute the normalizing denominator coefficients. In [15] the authors take the fast, normalized cross-correlation algorithm one step further by using rectangular basis functions to approximate the template image. The number of computations in the numerator will then be directly proportional to the number of basis functions used to represent the template image. Using a smaller number of basis functions to represent the template image will certainly reduce the computation but it may give a bad approximation of the template image, which would result in poor image alignment. In [81] the author uses a pipelined FPGA architecture to perform the normalized cross-correlation operation. This increases the computation speed significantly.

There have been various other improvements and implementations of the normalized cross-correlation algorithm in literature however none of the implementations, to our knowledge,

try to tackle the computational intensity problem of the normalized cross-correlation algorithm from an analog signal processing perspective.

2.3. Reasons for Choosing Normalized Cross-correlation

There are a lot of intensity based stereo correspondence algorithms. We chose Normalized Cross-correlation (NCC) as the algorithm that we would implement because of the following reasons:

1. The images being aligned in the patch-based approach have translation in the X and Y direction but no rotation or shear. NCC algorithm performs well for such images.
2. NCC is less sensitive to variation in the brightness and contrast of two images being aligned.
3. The NCC algorithm lends itself for analog computation.

2.4. The Normalized Cross-correlation Algorithm

Template matching is one of the simplest methods used for image alignment in a stereo correspondence algorithm. There are two images to be aligned. One image is called the template and the other image is called the reference. The template image is generally divided into blocks of smaller images. There is a tradeoff between the depth accuracy that can be achieved and computation that can be handled in a NCC algorithm. Increasing the number of blocks by reducing the block size increases the accuracy to which depth can be estimated but it also increases the number of computations required. There is also a limit to which the block size can be reduced. If the block size is made too small, then it might not have enough information to align with a matching block. Therefore, choosing an optimum template block size is important.

Each template block is shifted on top of the reference image and at each point a correlation coefficient is calculated. This correlation coefficient will act as a similarity metric to identify the closest matching blocks.

According to [61] the use of cross-correlation as a similarity measure is motivated by the Euclidean distance measure.

$$d^2(u,v) = \sum_{x,y} [r(x,y) - t(x-u, y-v)]^2 \quad (1)$$

Where *r* is the reference image, t is the template image and the summation is over (*x, y*). (*u,v*) represents the fact that the template image is being shifted by '*u*' units (pixels) in the Y (Vertical) direction and by '*v*' units (pixels) in the X (Horizontal) direction. At each shift, a distance is calculated between the reference image and template image.

Expanding equation (1) gives:

$$d^2(u,v) = \sum_{(x,y)} [r^2(x,y) + t^2(x-u, y-v) - 2r(x,y)t(x-u, y-v)] \quad (2)$$

The first two terms of (2) represent the energies of the reference image block and template image block respectively and these are constant terms. The similarity between the images is then determined by the cross-correlation term given by,

$$c(u,v) = r(x,y)t(x-u, y-v) \quad (3)$$

The disadvantage of using cross-correlation as a similarity measure is that it is an absolute value. Its value depends on the size of the template block. Also, the cross-correlation value of two exactly matching blocks may be less than the cross-correlation value of a template block and a bright spot. The way around this problem is to normalize the cross-correlation equation.

Equation (4) gives the Normalized Cross-correlation equation.

$$C(u,v) = \frac{\sum_{(x,y)}\left[r(x,y) - \bar{r}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\left\{\sum_{(x,y)}\left[r(x,y) - \bar{r}_{u,v}\right]^2\left[t(x-u,y-v) - \bar{t}\right]^2\right\}^{0.5}} \quad (4)$$

In the above equation $\bar{t}$ represents the mean of the template image block and $\bar{r}_{u,v}$is the mean value of the reference image present under the template image block. The two terms in the denominator of the above equation represents the variances of the zero-mean reference image and template image respectively. Due to this normalization, the correlation coefficient is independent of changes to image brightness and contrast.

Let the size of the reference image be R_x x R_y and let the size of each template block be T_x x T_y. The template image block is shifted *u* pixels in the Y direction and *v* pixels in the X direction and the normalized correlation coefficient is calculated for each value of (*u,v*) using equation (4). For calculating the numerator of equation (4) at each pixel shift, we need (T_x * T_y) multiplications and (T_x * T_y) additions. The total number of shifts required will be (*u***v*). This gives the number of operations per block to be (2**u***v***T*$_x$**T*$_y$). If we have a total of *M* x *N* template blocks the total number of computations required to calculate the numerator of equation (4) is (2**u***v***T*$_x$**T*$_y$) **M***N*. The denominator of equation (4) is also very computational.

2.4.1. Fast NCC Algorithm: Sum Table Method for Calculating Denominator

In [61] Lewis proposes an efficient method for calculating the denominator of equation (4). To simplify the calculation of the denominator a sum table method is used. The idea is to generate two sum tables *s*(*u,v*) and *s*^{2}(*u,v*) over the reference image *f*(*x,y*) and *f*^{2}(*x,y*). The sum table can be defined as follows,

$$s_1(u,v) = f(u,v) + s_1(u,v-1) + s_1(u-1,v) - s_1(u-1,v-1) \quad (5)$$

$$s_1^2(u,v) = f^2(u,v) + s_1^2(u,v-1) + s_1^2(u-1,v) - s_1^2(u-1,v-1) \quad (6)$$

Where $S_1(u,v)$ and $S_1^2(u,v) = 0$ for $u,v < 0$.

$$s_2(u,v) = f(u,v) + s_2(u,v-1) + s_2(u-1,v) - s_2(u-1,v-1) \tag{7}$$

$$s_2^2(u,v) = f^2(u,v) + s_2^2(u,v-1) + s_2^2(u-1,v) - s_2^2(u-1,v-1) \tag{8}$$

Where $S_2(u,v)$ and $S_2^2(u,v) = 0$ for $u,v < 0$.

Consider the terms in the denominator of equation (4),

$$\sum_{x,y}[r(x,y) - \bar{r}u,v\,]^2 \;=\; \sum_{x,y} r^2(x,y) \;+\; \sum_{x,y} r_{u,v}^2 - 2\bar{r}_{u,v}\sum_{x,y} r(x,y) \tag{9}$$

$$\sum_{x,y}[t(x-u,y-v) - \bar{t}\,]^2 = \sum_{x,y} t^2(x-u,y-v) + \sum_{x,y}\overline{t^2} - 2\bar{t}\sum_{x,y} t(x-u,y-v) \tag{10}$$

Using the sum table method, we only need 3 additions/subtractions to calculate each of the double summations in equation (9) and equation (10) and this is independent of the size of the template block selected. However, since we are taking an analog processing approach we can further reduce the computation by adopting a different approach.

2.4.2. Fast NCC Algorithm: Calculating the numerator

The numerator of the NCC equation can be expanded and simplified as follows,

$$N(u,v) = \sum_{x,y} r(x,y)t'(x-u,y-v) - \bar{r}(u,v)\sum_{x,y} t'(x-u,y-v) \tag{11}$$

Where, $t'(x-u,y-v) = t(x-u,y-v) - \bar{t}$

Since $t'(x,y)$ has zero mean, the second term in equation (11) reduces to zero. The numerator of equation (4) reduces to

$$N(u,v) = \sum_{x,y} r(x,y)t'(x-u, y-v) \quad (12)$$

According to the above equation the numerator of the Normalized Cross-correlation equation can be implemented as a 2D correlation operation. This can be implemented using 2D FFTs. However, the computational complexity of the numerator is still high. Analog signal processing approach can be an excellent fit to perform 2D correlation operations as will be demonstrated in later sections.

2.4.3. Modifications to the NCC algorithm

In a general normalized cross-correlation algorithm, the template image is divided into blocks and each block is shifted on top of the reference image. At each shift a normalized correlation coefficient is calculated. All the pixels in the block are used to perform this calculation. Once this is done for all shifts, a best matching block is picked and all the pixels in the template block are assigned the same shift/disparity value. However, this can be very computationally intensive. Following modifications are introduced to the general NCC algorithm to improve the computational efficiency.

2.4.4. Reducing the number of shifts

In the worst-case scenario where there is no information available about the camera system or the scene, a brute force approach has to be used where the template image blocks have to be shifted all over the reference image. The computational complexity in this case would be very high. When some information is available about the camera system the maximum disparity that will be observed can be calculated and hence the number of shifts can be restricted. From [56] we know that the relation between the depth and disparity can be written as,

$$D = \frac{Bf}{Nx} \tag{13}$$

Where, *D* is the depth or distance of the object from camera, *B* is the Baseline i.e. distance between the centers of two cameras, *f* is the focal length of the camera, *N* is the disparity in terms of pixel shifts and *x* is the Pixel size. Using this equation, we can calculate the maximum disparity that can be expected in a stereo camera system. However, this does not address the fact that the number of computations that must be performed per block for each shift is still high.

A pre-processing step that is generally used in most stereo correspondence algorithms is image rectification. Image rectification projects stereo images onto a common reference plane so that the correspondence points have the same row coordinates. This essentially transforms the 2D stereo correspondence problem to 1D. However, the rectification process itself will add to the computational complexity of the algorithm.

2.4.5. Using diagonal elements

In order to further reduce the computation and address the above-mentioned issues we decided to use only the diagonal elements of the template image block and the reference image blocks to compute the correlation coefficient. The thought behind this approach is that the diagonal elements of a block have the least spatial redundancy when compared to the neighboring horizontal and vertical pixels [1]. By introducing this modification, we have effectively converted the problem of 2D NCC operation to a 1D NCC operation. This is very similar to the image rectification operation but since we are choosing only the diagonal elements, we introduce two advantages both of which contribute to the reduction in computation.

1. We are not using an algorithm to reduce the NCC operation from 2D to 1D, it is a natural result of the data selection process and hence it does not involve any additional computations.

2. Since we are choosing only the diagonal elements, the number of computations per block is reduced to a great extent i.e. if we have a template image of size T_x x T_y the total computations per block in the numerator now reduces from (T_x x T_y) additions and multiplications to only T_x(=T_y) additions and multiplications per shift.

   This modification reduces the NCC equation to:

$$C(u,v) = \frac{\sum_{(x,y)}\left[r(x) - t\left(x-(u,v)\right)\right] - \bar{r}_{u,v}\bar{t}_{u,v}}{\left\{\sum_x\left[r(x) - \bar{r}_{u,v}\right]^2\left[t(x-(u,v)) - \bar{t}_{u,v}\right]^2\right\}^{0.5}} \quad (14)$$

The first term in the numerator is a 1D correlation operation which can be implemented efficiently in analog hardware. The second term in the numerator is calculating the mean and the denominator is calculating the variance. If implemented directly it takes two passes over the data to compute these values which increases the computation time. This problem was solved by introducing another modification to the NCC algorithm. We borrow from the field of statistics a one pass formula to efficiently calculate the variances [98]. Let $x_1, x_2 \ldots x_n$ be a set of real numbers for which the variance has to be calculated. Then we can define two quantities as,

$$A_K = \begin{cases} x_1, & k = 1 \\ A_{k-1} + \dfrac{x_k - A_{k-1}}{k}, & k = 2, \ldots, n \end{cases} \quad (15)$$

$$B_K = \begin{cases} 0, & k = 1 \\ B_{k-1} + \dfrac{(k-1)(x_k - A_{k-1})^2}{k}, & k = 2, \ldots, n \end{cases} \quad (16)$$

The variance can then be calculated as B_n/n.

2.5. Per-Pixel Stereo Correspondence Algorithm

The patch based stereo correspondence algorithm discussed in the previous section produces very coarse depth maps. This approach is suitable for applications where speed is more important than accuracy. However, in some of the other applications, a more accurate or a finer depth map is required. In these cases, a per-pixel approach of finding depth is preferred to a patch-based approach. In this section, we discuss a per-pixel stereo correspondence algorithm. The similarity measure used here is Sum Absolute Difference (SAD). We show that using an optimum overlap between blocks can give a similar performance as a per-pixel algorithm while significantly reducing the computational load. A hardware architecture is proposed which can further speed up computation.

2.5.1. Review of related work

There are some major challenges associated with recovering an accurate depth map like the presence of occluded pixels, noise in images and depth discontinuities. A multitude of stereo correspondence algorithms have been developed which address some or all of these challenges. In [89] the authors provide a thorough analysis and overview of the different stereo correspondence algorithms. One of the algorithms that has a good performance is proposed by Klaus et al. [57]. They use a segment-based stereo correspondence algorithm which uses belief propagation and a self-adapting dissimilarity measure.

2.5.2. Disparity Estimation using overlapping blocks

Most of these per-pixel stereo correspondence algorithms calculate a coarse depth estimate of the scene using local window-based approaches as the first step. One of the major problems associated with using the patch-based approach for calculating the coarse depth

estimates is that the algorithm does not perform well when there are depth discontinuities in a block. Also depending on the application, the estimated depth may be too coarse.

In order to obtain a finer depth estimate, and reduce the effect of depth discontinuities, overlapping blocks can be used. The higher the overlap, lesser is the effect of depth discontinuities and a finer depth map is produced. If the blocks are completely overlapping (except for one pixel) then this method can be thought of as per-pixel depth estimation. The drawback of using such an approach is that computation increases with the amount of overlap. The computational intensity of such an algorithm is higher than the patch-based approach and the digital system will be bogged down by the high computational load. In order to get around this problem, we can use analog signal processing. The similarity measure being used here is Sum Absolute Differences (SAD) as shown in the equation below.

$$SAD(u,v) = \sum_{n=1}^{blkHeight} \sum_{m=1}^{blkWidth} |R(u,v) - T(u+n, v+m)| \qquad (17)$$

2.6. Hardware Architecture

In a standard CMOS image sensor, there are photodiodes that produce electrons proportional to the amount of light intensity that strikes them. This is then converted into voltage levels which are read out by the readout circuitry. To remove noise, a process called correlated double sampling (CDS) is used. After this, the signals are amplified. All this happens in the analog domain. These signals are then digitized using analog-to-digital converters (ADCs) and stored in memory or are sent to digital processors for further processing. The signal for the most part is in the analog domain and we can utilize this to our advantage to perform analog processing.

We have come up with a new imaging architecture which would best utilize the features of both analog and digital domains. Figure 2 shows a top-level block diagram of the proposed

architecture. In this architecture, we have the digital system accessing the sensor and it is pre-processing, digitizing and storing the images in memory as before. However, we now have an analog system that is accessing the analog data on the sensor, processing it and then feeding it into a digital machine for any further computations. By doing this we have separated the process of image acquisition which is being done by a digital system and image processing which is being done by an analog system. The biggest advantage of such an architecture is that they are operating in parallel i.e. the image acquisition is independent of the processing. This is not true in the case of completely digital systems. In a fully digital system the image processing operation cannot start until the images have been completely acquired and stored in memory. In this hybrid system, the analog block is performing the computationally intensive task of running the stereo correspondence algorithm as and when the image data is being read off the sensor.

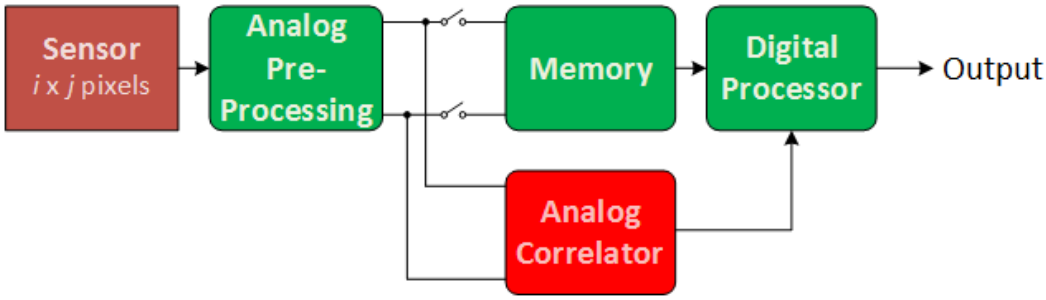


Figure 2: Block Diagram of the proposed hardware architecture

This means that by the time the digital system has acquired the images, the analog processor would have finished its computation and the outputs will be ready to be used by the digital system.

Another important point to be noted is that the analog processing block is not in the signal path but in the control path. One of the biggest disadvantages of an analog system is the amount of noise added by it. However, in this kind of an architecture the analog block is not responsible

for signal acquisition and hence the problem of signal being corrupted by noise vanishes immediately.

2.6.1. Implementing the modified NCC algorithm in hardware

Consider the first term in the numerator of equation (14) which represents a 1D correlation operation. Figure 3 shows the implementation of the modified NCC algorithm in the new imaging architecture. We have a digital system which is accessing the sensor data, pre-processing, digitizing and storing it in memory. We have N analog channels which read the analog data from the sensor directly. These N analog channels can be grouped into pairs in which one channel is used to read the reference image data and the other channel is used to read template image data. The CMOS image sensors are capable of accessing individual pixel data. The readout circuitry is used to selectively read the diagonal elements of template and reference image blocks. Once the analog data has been read from the sensor, we have it available to perform the NCC algorithm. This is then fed to the multiplier and integrator which together perform the correlation operation. The outputs are digitized and sent to the image signal processor (ISP) for further processing. This calculates the numerator of the NCC algorithm. For calculating the denominator and other numerator terms the one pass method shown in equations (15) and (16) can be used. This can be done efficiently by a digital system.

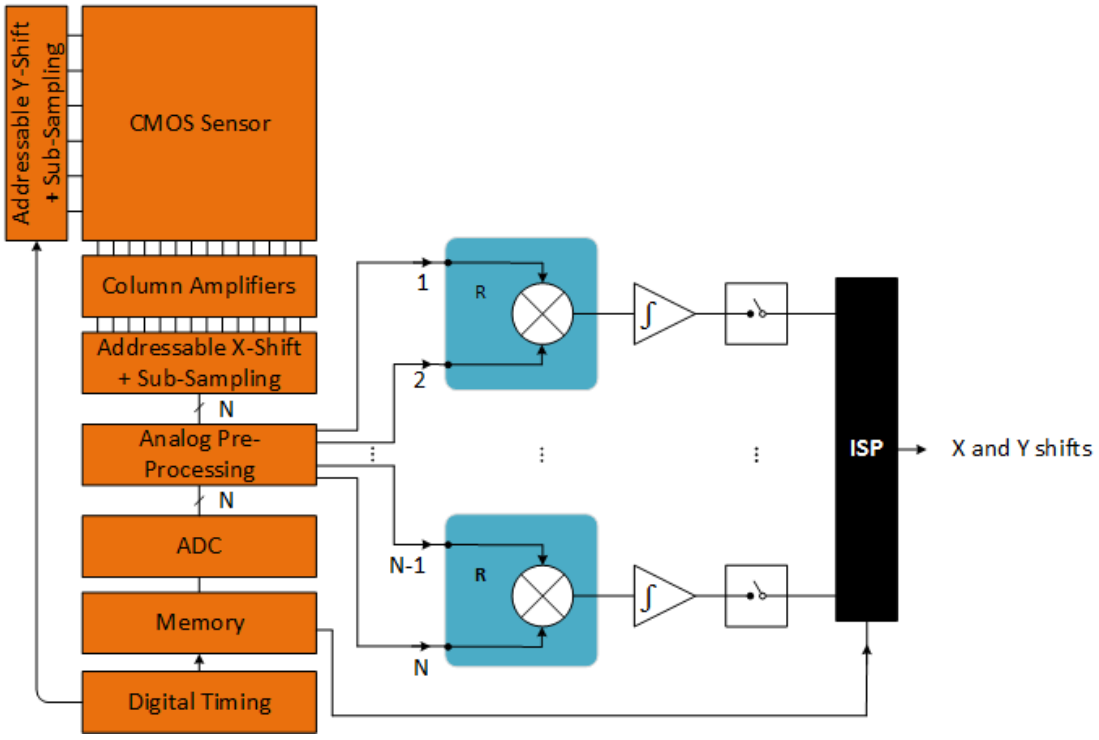


Figure 3: Hardware architecture for implementing the modified NCC algorithm

2.6.2. Implementing the SAD algorithm in hardware

Figure 4 shows the implementation of the SAD algorithm in a very similar hardware architecture. Entire rows or columns of the reference and template image data are read at once and their difference is calculated. This data is then passed through an absolute value circuit and then through the integrator. The integrator accomplishes a 1-D summation. To perform the second dimensional summation, the channels are grouped, and analog summers can be used. The outputs of these summers are sampled. These values are then fed into an ISP where any further processing and decision making that is required is performed. As before, the image acquisition is independent of processing and they are happening in parallel. This leads to a significant reduction in computation time.

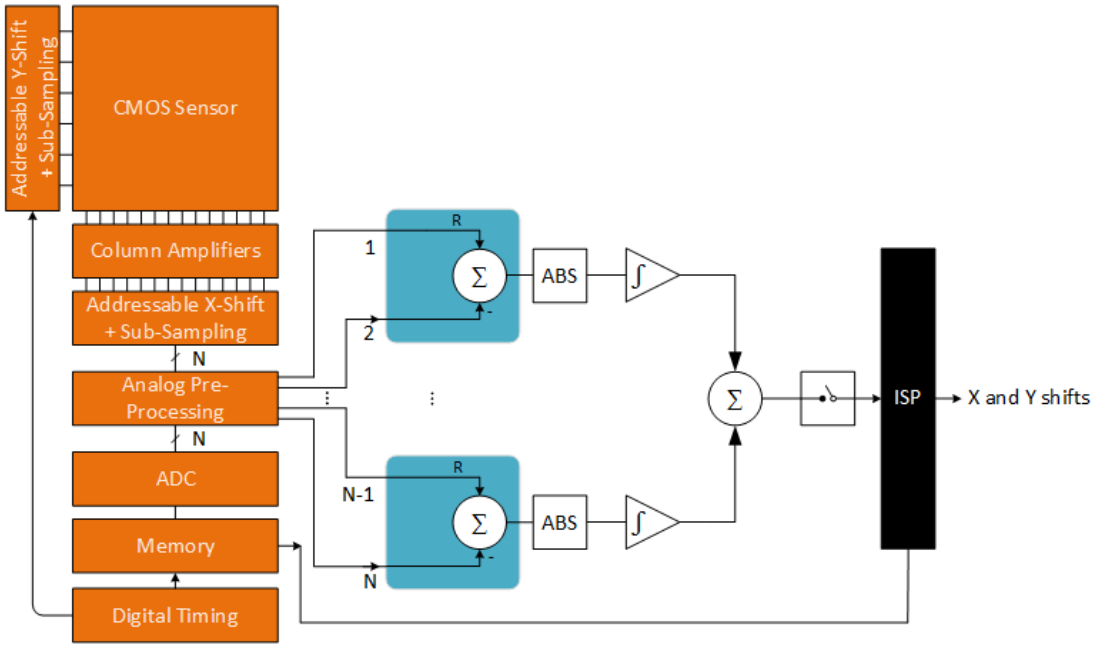


Figure 4: Hardware architecture for implementing the SAD algorithm

2.7. Stereo Correspondence Algorithm Results

This section presents the simulation results for both the Patch-based stereo correspondence algorithm for which NCC was used as a similarity measure and the per-pixel based stereo correspondence algorithm for which SAD was used as a similarity measure. All simulations have been performed in MATLAB.

2.7.1. Patch Based Stereo Correspondence Algorithm

In this section, we compare the performance of the modified NCC algorithm to the original algorithm to show that the modified algorithm is faster and has a performance similar to the original algorithm. Since the modified algorithm has been developed to be implemented in analog hardware various other simulations are run that measures the performance of the modified algorithm.

We have considered 15 sets of unrectified, grayscale stereo image pairs on which simulations have been performed. These images have been captured under different illumination conditions which include incandescent light (In_Incd), outside bright light (Out_brt), outside low light (Out_clds), outside mixed shade lighting (Out_mxdshd). This allows us to test the performance of the algorithm in a more robust manner.

2.7.1.1. Evaluating the performance of the modified NCC algorithm

The performance of the image alignment algorithms can be evaluated in a variety of different ways. Here the correlation coefficient is used as a performance measure. Once the final disparity values for all the template blocks are obtained, each template block is shifted by the disparity values obtained for that block. In order to get a uniform disparity variation across the entire image an interpolation technique is used. At the end of this process the two stereo images have been aligned. The correlation coefficient is calculated between the two aligned images and it is used as an indicator of the performance of the algorithm.

Figure 5 shows a comparison of the correlation coefficients obtained for 15 stereo image pairs by using the original NCC algorithm and the modified NCC algorithm. Each stereo image pair has a size of 1080x1920 pixels. The template block size chosen here is 128x128 pixels. As can be seen from the figure, the performance of the modified NCC algorithm is very close to the original NCC algorithm. The performance was also tested for various other template block sizes and uniform performance was obtained for all. A speedup of 2x in MATLAB run time was observed for the modified NCC algorithm over the original algorithm.

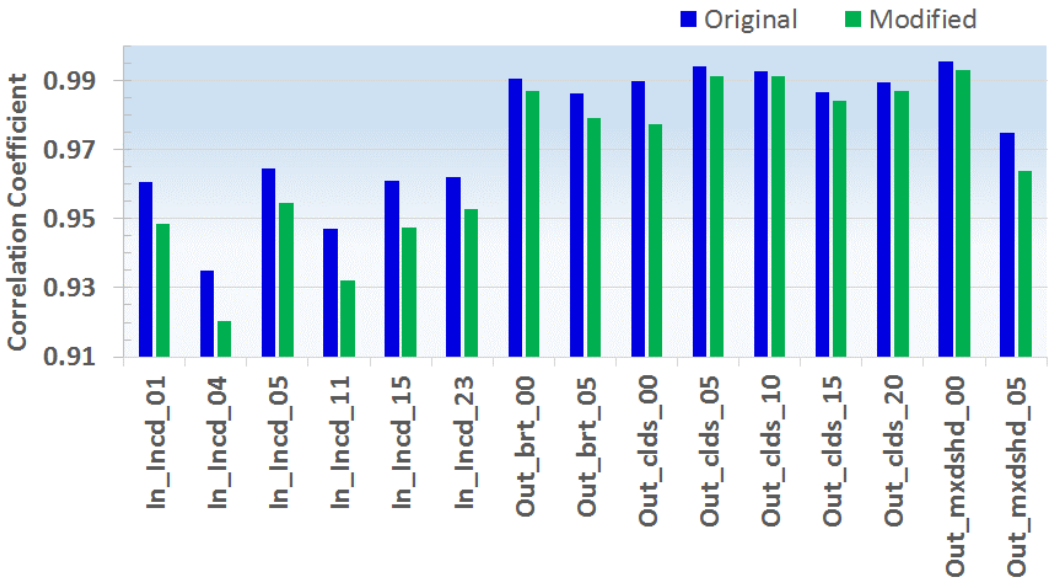


Figure 5: Performance comparison of the original NCC algorithm to the modified NCC algorithm

Table 1 shows the fifteen stereo image pairs used for simulation and their correlation coefficient before aligning them. The second column shows the correlation coefficients calculated for these image pairs after aligning them using the modified NCC algorithm. The third column shows the percentage improvement in the correlation coefficient after alignment. As can be seen from the table, the percentage improvement is not uniform. The percentage improvement depends on the initial misalignment of the stereo image pairs. Also, these results are for one pass of the algorithm. Multiple passes of the algorithm can be performed by continuously reducing the size of the template blocks which is called a pyramidal approach.

As an example of the performance of the modified NCC algorithm, Figure 6 and Figure 7 are shown. Figure 6 shows an overlap of a pair of stereo images before alignment. The areas of magenta and green show the areas of misalignment between the two images. The correlation coefficient measured for these two images before alignment is 0.7247. Figure 7 shows the overlap of two images after aligning them using the modified NCC algorithm. As can be seen from the

figure there is hardly any misalignment between the two images. The correlation coefficient observed in this case is 0.9923.

Table 1: Correlation coefficients for 15 stereo image pairs before and after alignment and the percentage improvement

Stereo Image Pairs	Correlation Coefficient before alignment	Correlation Coefficient after alignment using modified NCC	Percentage improvement in correlation coefficient
Inside_Incandescent_01	0.7678	0.9485	18.07
Inside_Incandescent_04	0.8403	0.9205	8.02
Inside_Incandescent_05	0.9475	0.9547	0.72
Inside_Incandescent_11	0.8643	0.9320	6.77
Inside_Incandescent_15	0.9091	0.9473	3.82
Inside_Incandescent_23	0.9164	0.9526	3.62
Outside_bright_00	0.6378	0.9868	34.9
Outside_bright_05	0.5221	0.9793	45.72
Outside_clouds_00	0.7231	0.9774	25.43
Outside_clouds_05	0.7245	0.9914	26.69
Outside_clouds_10	0.8218	0.9913	16.95
Outside_clouds_15	0.8105	0.9840	17.32
Outside_clouds_20	0.6607	0.9869	32.62
Outside_mixedshade_00	0.8677	0.9930	12.53
Outside_mixedshade_05	0.8941	0.9638	6.97

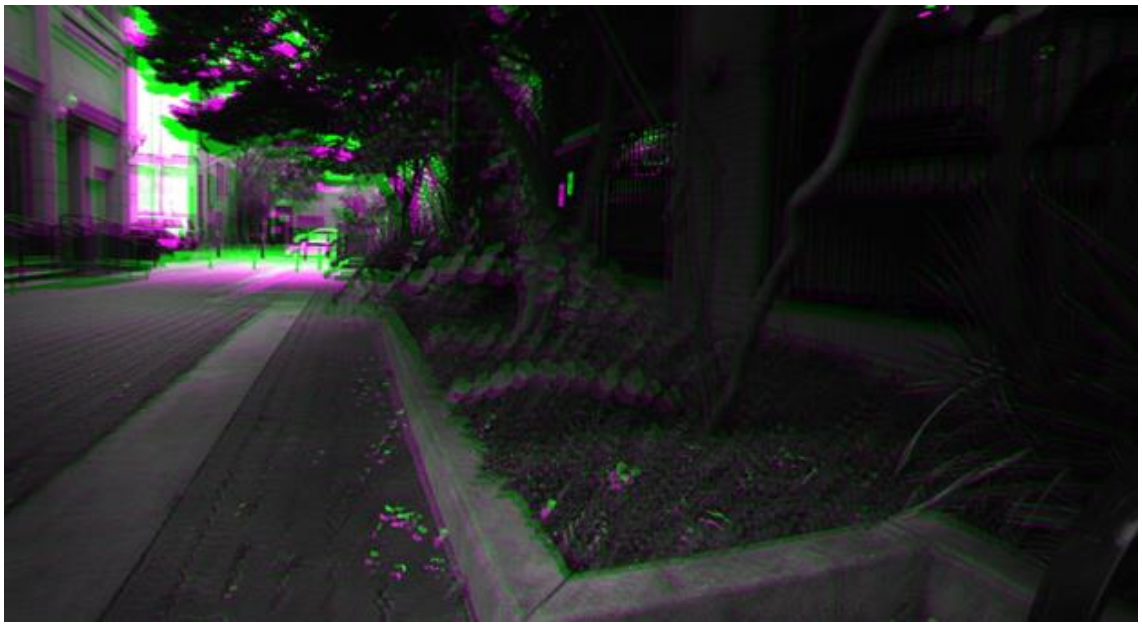


Figure 6: Overlap of a stereo image pair before alignment

Figure 7: Overlap of stereo image pair after alignment using modified NCC algorithm

Figure 8 and Figure 9 shows the disparity map for the image shown in Figure 6 and Figure 7 in X and Y direction respectively. These disparity values have been obtained through the modified NCC algorithm. The disparity values have been color coded and the color bar indicates the different disparity values. The disparity values vary from 18 to 33 for Figure 8 in the X (horizontal) direction. The disparity values vary from 43 to 53 for Figure 9 in the Y (vertical) direction.

2.7.1.2. Measuring the Robustness of the modified NCC algorithm

We know that the NCC algorithm is robust to changes in the intensity values of the images. Here, we measure the robustness of the modified NCC algorithm to changes in intensity values by changing the intensity values of one of the two stereo images. In the first case, we reduced the intensity values of the template image by 90% uniformly across the image and used the modified NCC algorithm to align the images. This analysis addressed the fact that the illumination of a scene

might change between the capture of two stereo images and the change in illumination was assumed to be uniform.

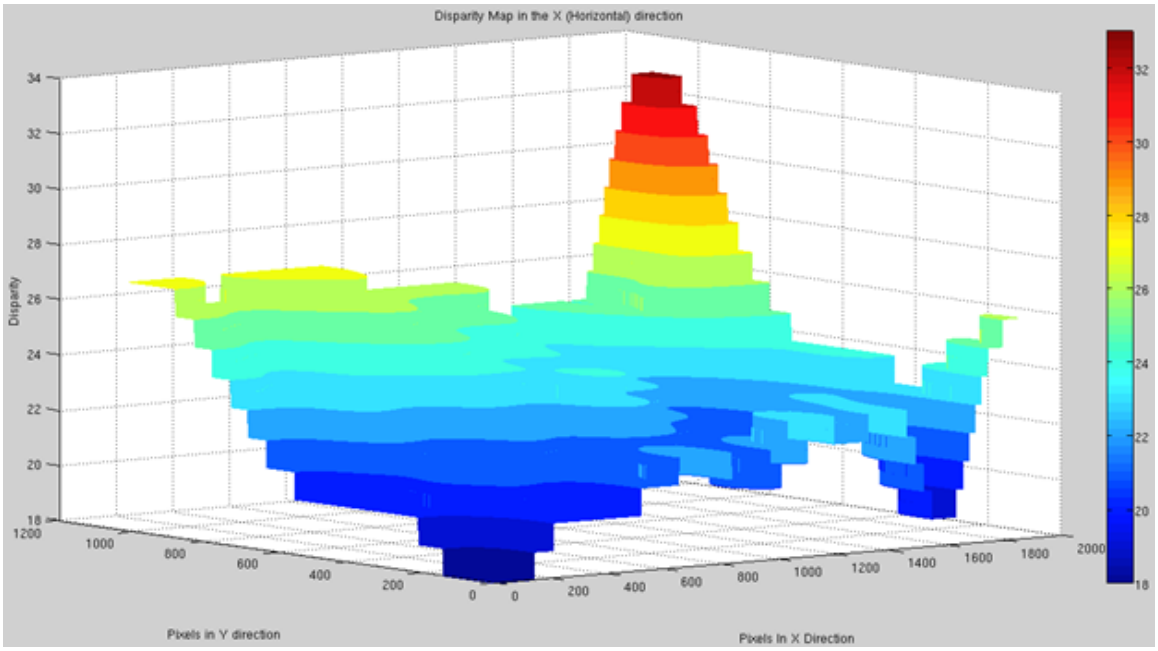


Figure 8: Disparity map in the horizontal (X) direction for the stereo image pair shown above

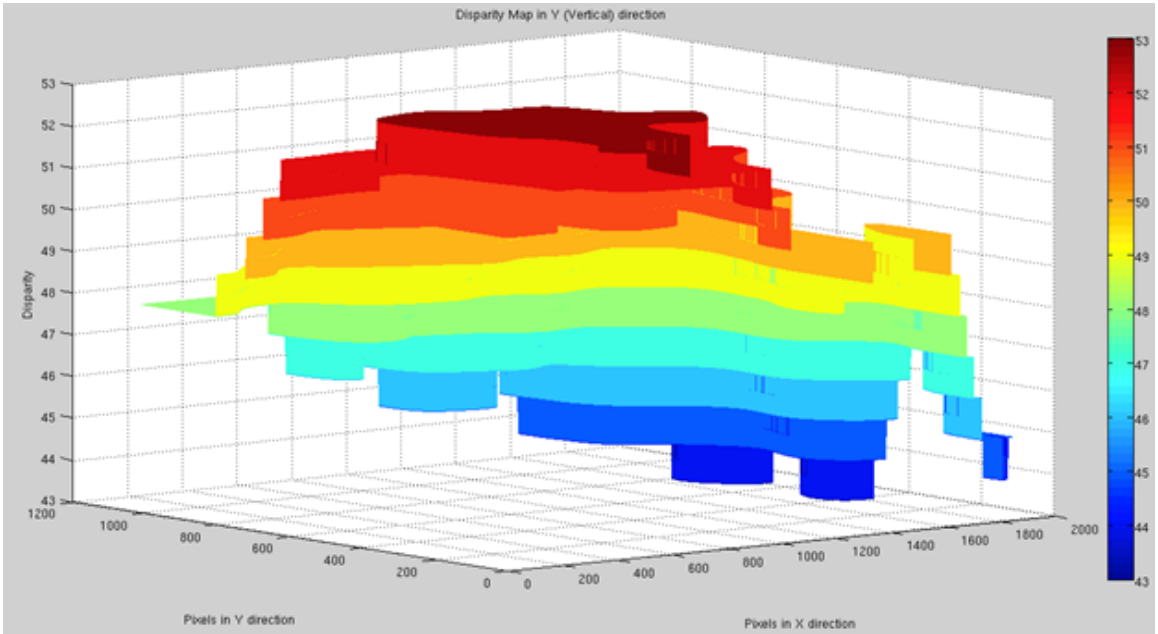


Figure 9: Disparity map in the vertical (Y) direction for the stereo image pair shown above

However, there might be rare situations where illumination on parts of the scene varies between captures of two stereo images. To address this issue, we randomly varied the intensity values of the template image and analyzed the performance of the modified NCC algorithm under these conditions as well. Figure 10 shows the results of these analysis. As it can be seen, the modified NCC algorithm is very robust to changes in the image intensity values.

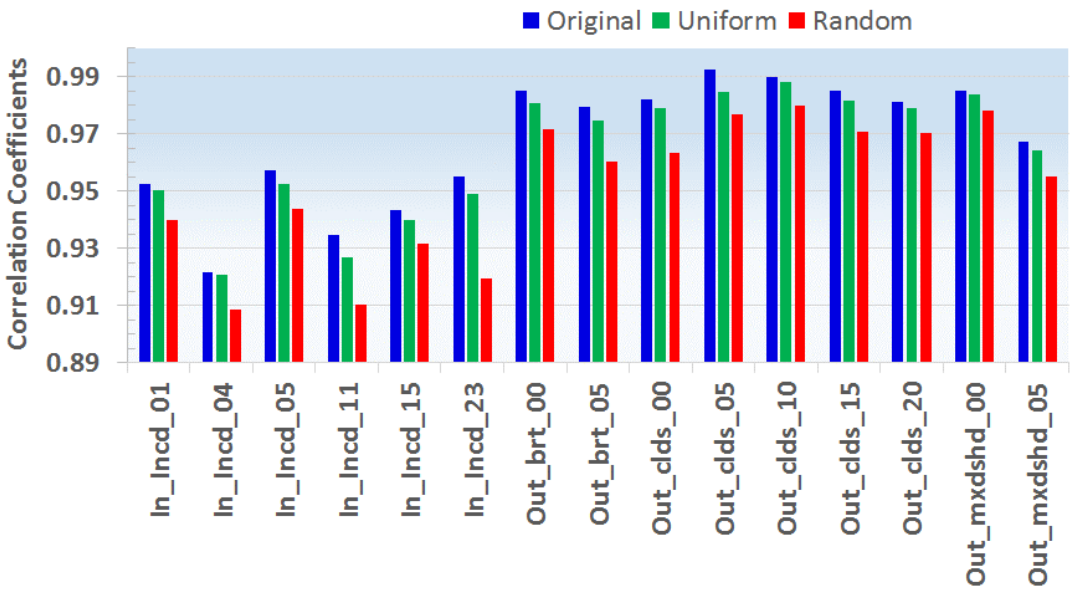


Figure 10: Robustness of the modified NCC algorithm to changes in image intensity values

The worst-case performance was observed for "In_Incd_23" stereo image which had a reduction in correlation coefficient of 3.7 %.

Since the algorithm has been developed to be implemented in analog hardware it is very important to characterize the performance of the algorithm in the presence of noise. The two analog circuits that have been considered to be the primary contributors of noise are the multiplier and the integrator. To simulate the addition of noise by analog circuitry we first find the RMS value of the image intensity values. We multiply this RMS value by a number which indicates the percentage of noise being added by the circuit. This value is then multiplied by a random number picked from a Gaussian distribution. The outcome of this process is a noise value which

is then added to the original image intensity. In our simulations, it was found that the algorithm is more sensitive to the noise added by the multiplier than that by the integrator. Hence, we maintain the noise added by the integrator at 20% and vary the amount of noise added by the multiplier. Figure 9 shows this performance variation. We have shown the performance for 3 different noise values added by the multiplier, 1%, 10% and 20%. The correlation values have been averaged over 100 runs. This shows that the algorithm is very stable in the presence of added noise by the analog circuitry.

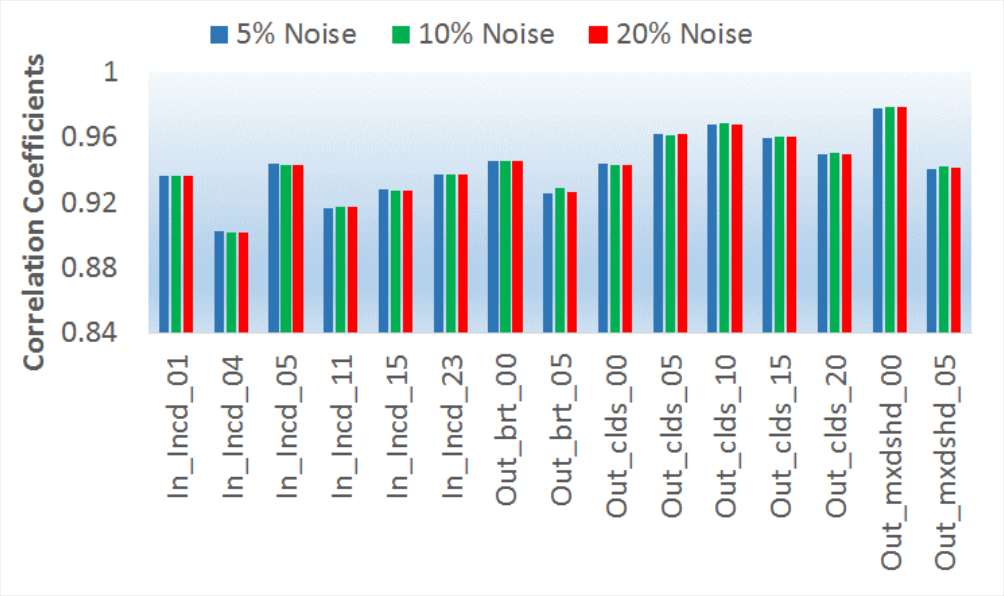


Figure 11: Performance of the modified NCC algorithm in the presence of Noise

2.7.1.3. A note on computation

The modifications proposed to the NCC algorithm contribute to a significant reduction in the computation of the algorithm. Simulation results show a 50% reduction in MATLAB run time for the modified NCC algorithm over the original algorithm. However, the actual reduction in computation is a factor of T_x which is the size of the template image block. The other factors which add to the reduction of computation time are the novel imaging architecture and analog processing. In the new imaging architecture, the analog processor works in parallel with the digital

acquisition system and hence it does not have to wait for the entire image to be acquired before the processing starts. By the time the acquisition is done the analog processor would have finished its computation. So, the image acquisition time can also be added towards the reduction in computation time. The implementation of the NCC algorithm is being done in analog hardware. The analog processor is not limited by the data converters or logic delays. The settling times of well-designed analog circuits are very small. Hence an analog implementation of the NCC algorithm would be faster than a digital implementation and would contribute towards a further reduction in computation time.

2.7.2. Per-Pixel Stereo Correspondence Algorithm

In this section, we test the performance of the SAD algorithm by performing various simulations in MATLAB. Since the SAD algorithm will be implemented in analog hardware we have considered the performance of the algorithm in the presence of added noise by the analog circuitry. We have run simulations on two stereo images from the standard Middlebury dataset [43]

The performance of the image alignment algorithms can be evaluated in a variety of different ways. Here the performance metrics used are correlation coefficient and RMS disparity error. Once the final disparity values for all the template blocks or pixels are obtained, each template block or pixel is shifted by the disparity values obtained. At the end of this process the two stereo images have been aligned. The correlation coefficient is calculated between the two aligned images and it is used as an indicator of the performance of the algorithm. The Middlebury dataset also has a ground truth disparity map available. This is used to compute the RMS disparity error between the ground truth disparity map and the computed disparity map which can also be used as a performance measure as suggested in [89].

Figure 12 and Figure 13 shows the correlation coefficients obtained for the Middlebury dataset ‘Baby1’ and ‘Bowling1’ respectively. We have measured the correlation coefficient for 4 different block sizes 5x5, 7x7, 11x11 and 15x15 pixels. The overlap between the blocks is varied for each block size, from 0 pixels which represents a patch-based approach to n-1 pixels which represents the per-pixel approach, where n is the size of the block. The x-axis in Figure 12 and Figure 13 represents the pixel overlaps. For each block size and overlap we compare the performance of the SAD algorithm with and without noise being added by the analog circuitry. The amount of noise added by the analog circuits is fixed at 5% of the RMS signal level. The procedure for adding noise is the same as explained in section 2.7.1.2. As can be seen from the figures the performance of the algorithm is good, even in the presence of noise.

Figure 14 and Figure 15 shows the comparison of the RMS disparity error between the ground truth disparity map and the computed disparity map for the Middlebury dataset ‘Baby1’ and ‘Bowling1’ respectively for 4 different block sizes. The overlap between blocks is also varied as before which is represented by the x-axis of Figure 14 and Figure 15. We compare the performance of the SAD algorithm both with and without the noise added by the analog circuitry. The amount of noise added is 5% of the RMS signal level. As can be seen from the figures the performance of the algorithm after the addition of noise is very close to the ideal case.

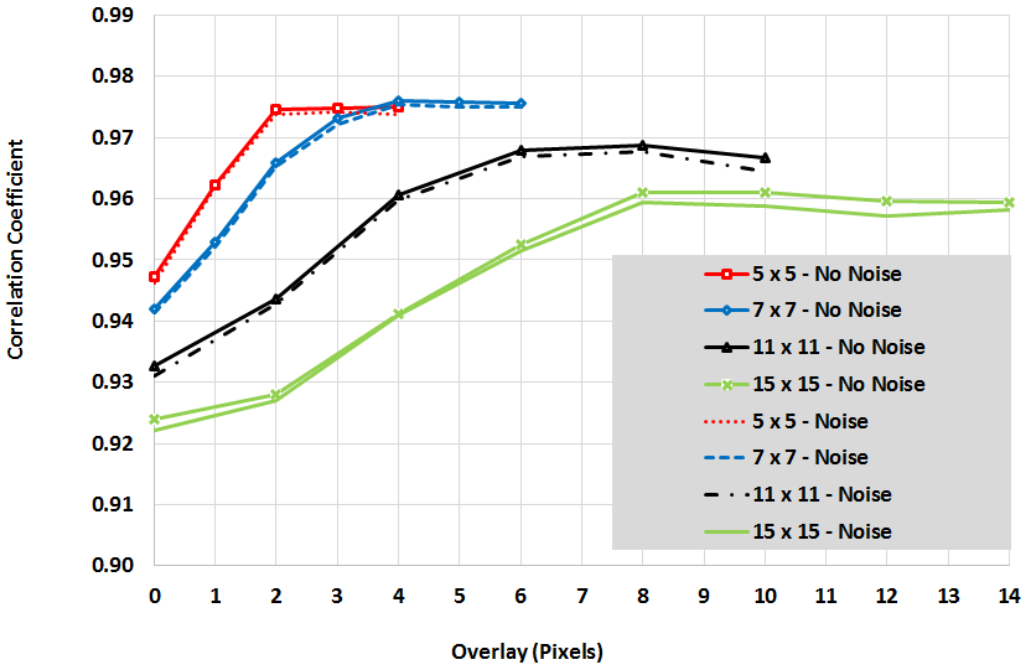


Figure 12: Correlation Coefficient for different block sizes with and without noise (Dataset: 'Baby1')

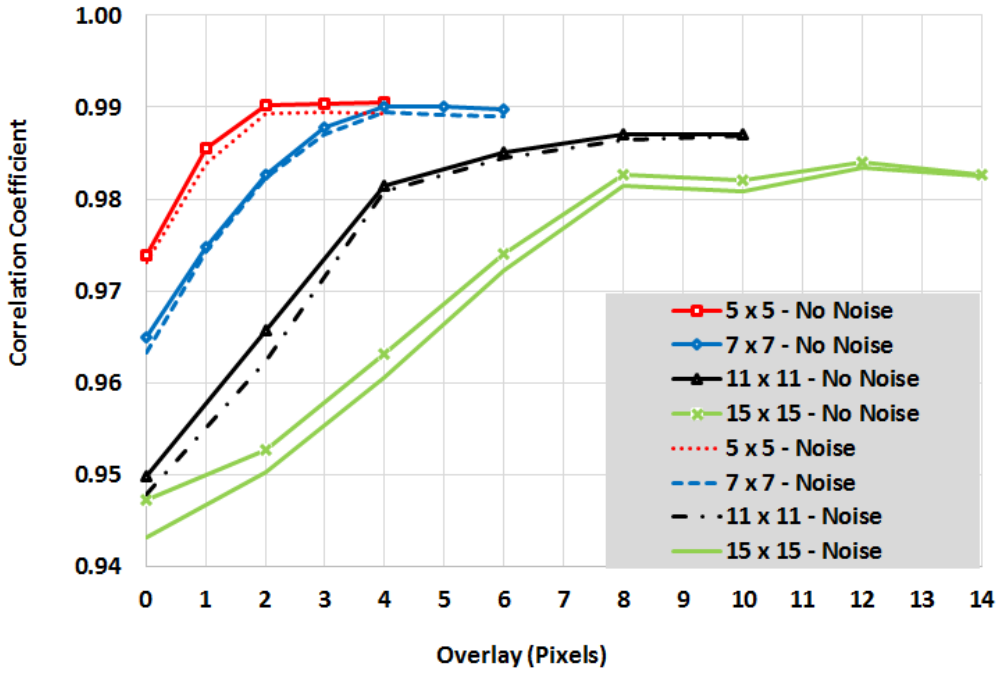


Figure 13: Correlation Coefficient for different block sizes with and without noise (Dataset: 'Bowling1')

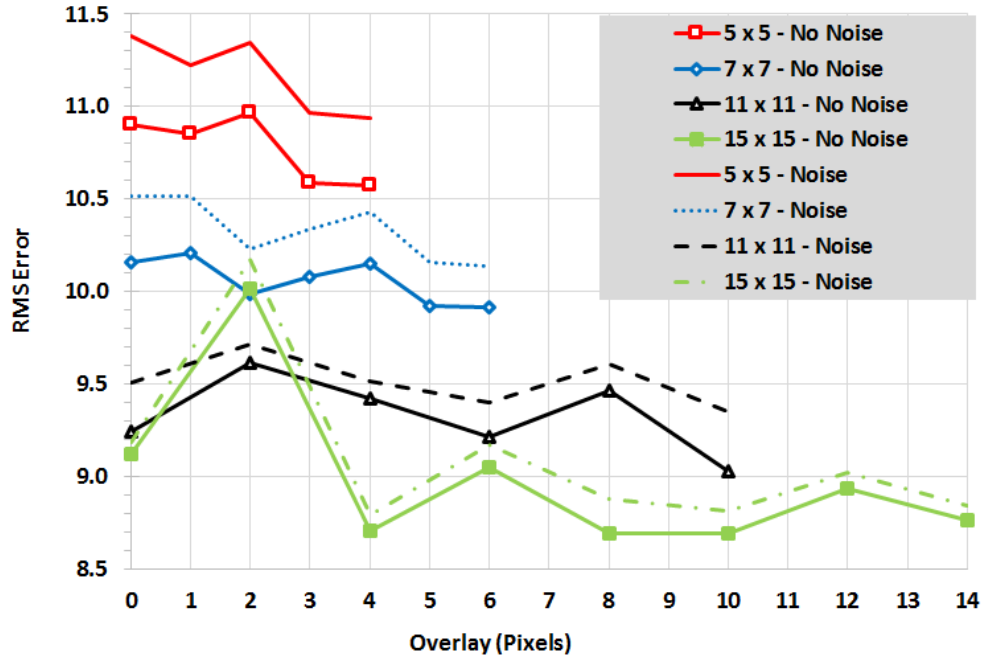


Figure 14: RMS disparity error for different block sizes with and without noise (Dataset: 'Baby1')

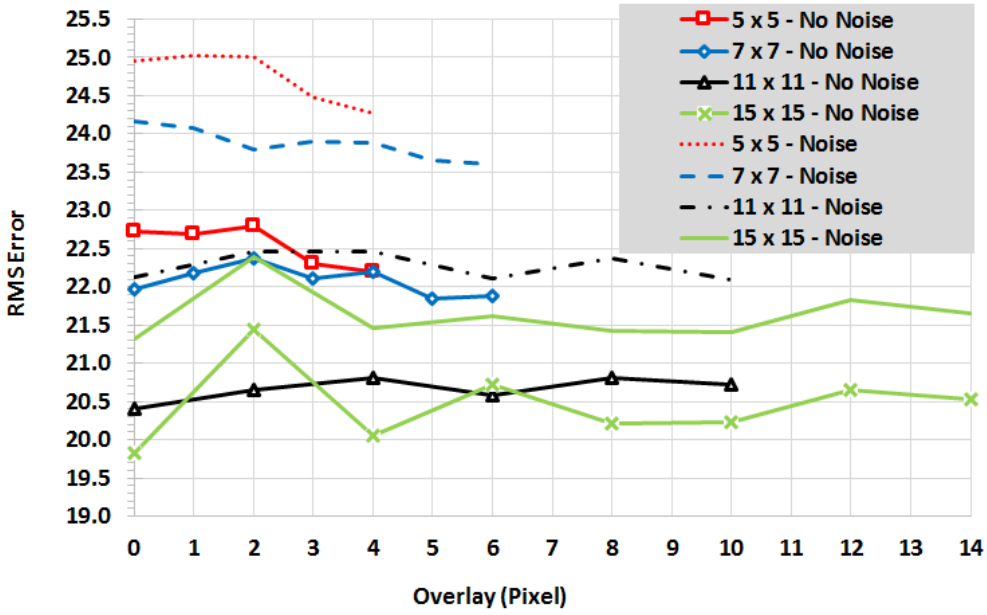
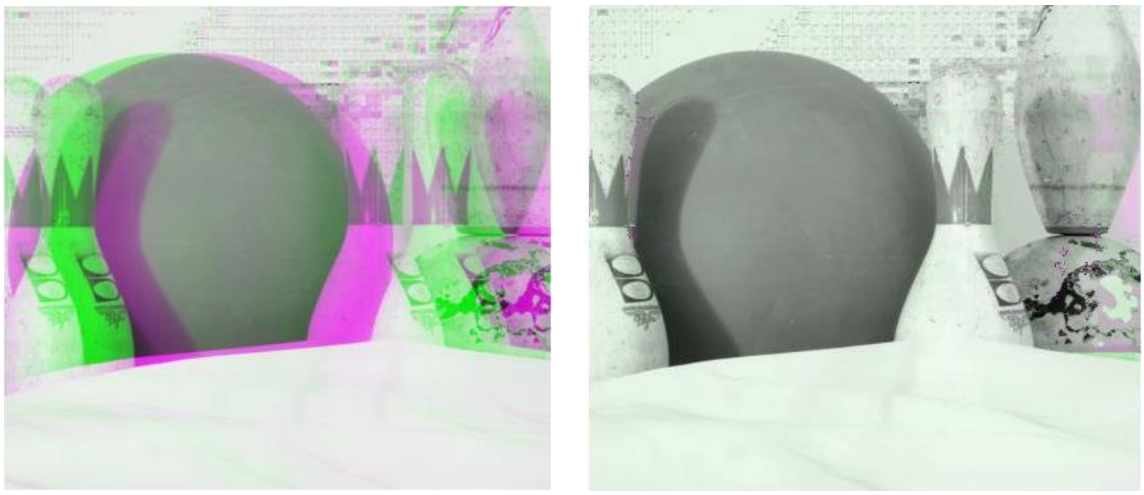


Figure 15: RMS disparity error for different block sizes with and without noise (Dataset: 'Bowling1')

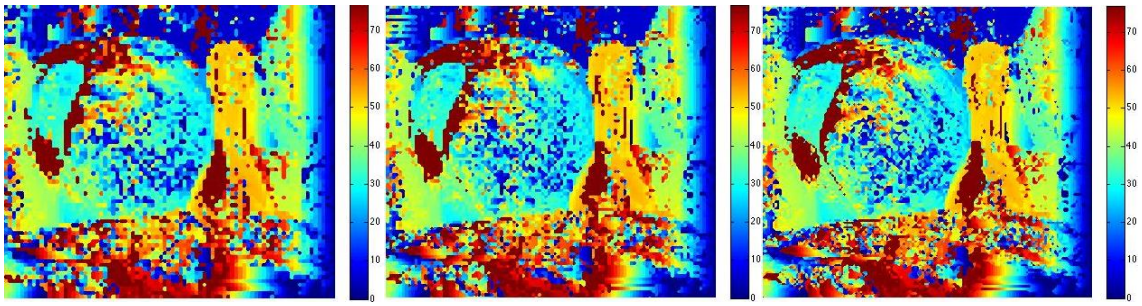


(a) (b)

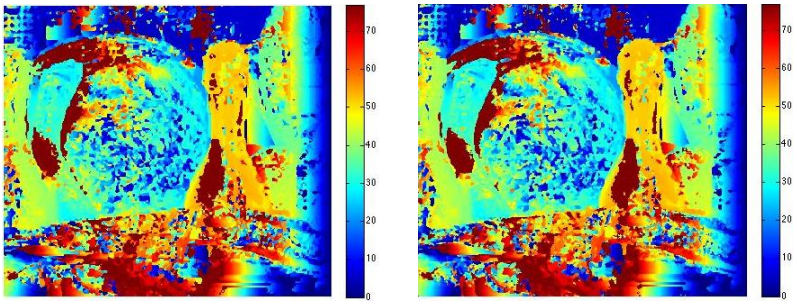
Figure 16: (a) Overlap of stereo images before alignment (b) Overlap of stereo images after alignment (Dataset: 'Bowling1')

Figure 16 shows the overlap of two stereo image pairs before and after alignment. Before alignment the correlation coefficient is 0.5189 and after alignment the correlation coefficient is 0.9906.

Figure 17 shows the x-disparity map obtained for the 'Bowling1' dataset for a window size of 5x5 pixels for different overlaps. The figures (a), (b), (c), (d) and (e) show disparity maps with 0, 1, 2, 3 and 4 pixel overlaps respectively. The disparity map has been generated after considering a 5% noise added by the analog circuitry during processing. As can be seen from the disparity map, as the overlap increases the disparity map generated gets finer, but it hits a saturation level at a certain point after which we do not see a lot of improvement. This is also evident from the correlation coefficient curves in Figure 14 and Figure 15. This shows that we have to select an overlap factor which gives a good performance while not adding to unnecessary computation. We believe an overlap factor of 60% to 70% would be a good trade-off between computation and accuracy of the disparity map. By reducing the overlap, the amount of computation is also reduced significantly.



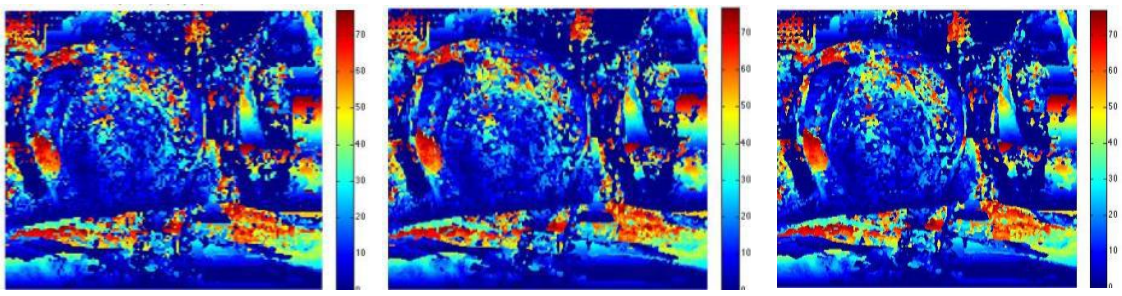
(a) (b) (c)



(d) (e)

Figure 17: X disparity map for with different overlaps for a window size of 5x5 pixels (Dataset: 'Bowling1')

Figure 18 (a), (b) and (c) shows the disparity map in the y direction for the 'Bowling1' dataset for a window size of 5x5 pixels and 3 different overlap settings of 2, 3 and 4 pixels. The images in the Middlebury dataset are rectified but as can be seen from the figure there are some variations in the y direction which were not accounted for in the rectification process. This may be due to pixel occlusions [84]. The advantage of performing a coarse alignment in the analog domain is that the operation is being performed on raw image data i.e. unrectified images. Hence, we calculate disparities in both x and y directions. This gives us extra information about the pixel disparities which cannot be obtained by performing stereo correspondence on rectified stereo images.



(a) (b) (c)

Figure 18: Y disparity map with different overlaps for a window size of 5x5 pixels (Dataset: 'Bowling1')

2.8. Conclusion

In first part of this dissertation we propose analog signal processing as a solution for handling the high computational load of some of the stereo correspondence algorithms while simultaneously meeting the reduced SWaP requirements. The analog processor will be used to augment the digital processor and work in parallel with it to perform key computations, making the system faster and more efficient. We implement two highly computational stereo correspondence algorithms to align stereo image pairs. Novel modifications were proposed to both the NCC algorithm and SAD algorithm which reduced the computation and made the algorithm efficiently implementable in analog hardware. The modified NCC algorithm has a 50% reduction in MATLAB run time over the original algorithm whereas the modified SAD algorithm has a speed up of more than 4x in MATLAB run time over the original algorithm. The SAD algorithm produced a finer depth map. The actual analog hardware implementation of the algorithm and the new imaging architecture will contribute to a further reduction in computation time as compared to a digital implementation. Various other simulations were also run to check the robustness and performance of the algorithm. The experimental results obtained are very promising and we believe analog processing will be a viable solution to these problems.

CHAPTER 3

ANALOG SIGNAL PROCESSING FOR MACHINE VISION

Most of the machine vision algorithms involve performing low-level and mid-level image processing tasks like image enhancement, filtering, segmentation and classification of objects. The accuracy achieved by analog processing is often sufficient for these image processing tasks. Another common feature in these algorithms is the requirement for parallel processing. For example, most of the low-level and mid-level image processing tasks can be implemented through convolution of pixel intensities with filter kernel values. In these cases, each image pixel can be processed independently and in parallel. Hence, analog processing is a suitable candidate for these tasks.

There have been successful attempts to develop analog vision chips that are faster than digital systems using a variety of technologies ranging from CMOS devices([31], [47], [50], [85], [88], [109], [110]) to the more recently developed spintronic based Nano devices [69]. Neuromorphic computing has gained a lot of popularity in the recent years as a possible solution method for some of the machine vision algorithms ([29], [34], [36], [37], [48], [54], [80], [93]). Some of the vision chips described in these research papers were designed as application specific integrated circuits while other papers show simulation results. The design of application specific integrated circuits is a long and expensive process. Simulation results are a good indicator of performance but cannot always indicate actual circuit performance due to many constraints, and the results also depend on how well the circuit models have been defined.

In this work, we have taken a different approach. We have built a prototype of an analog processing board using commercial off-the-shelf (COTS) components. The advantages of developing such a prototype board include flexibility, cost savings, decreased time-to-market and

lower risk than a direct custom integrated circuit design. It is also better than simulating with circuit models alone. It will help us understand analog processing techniques required for machine vision applications and create better circuit architectures for an integrated circuit design. The board has three analog channels which can be used to perform various operations such as convolutions and correlations in the analog domain. In this part of the dissertation, we focus on providing experimental results that demonstrate the functionality of the analog processing board. We also propose the development of an integrated circuit design called the Analog Space-Time Convolution Processor (ASTCP) based on a similar architecture and compare the performance of the proposed analog hardware accelerator against existing digital accelerators and other analog processors reported in literature. Finally, we report a case study where we use the processor for an object detection and recognition application and show that the processor has excellent performance.

3.1. Background

The availability of various digital hardware in the market like General Purpose Graphics Processing Units (GPGPUs), Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs) have led to the implementation of digital hardware accelerators for machine vision applications [55]. While the performance of these digital hardware accelerators is impressive, their biggest drawback is the power consumption and large size. The alternative is to use analog hardware accelerators in conjunction with digital accelerators.

The feasibility of analog computation for image processing applications was demonstrated as early as 1989 by Y. Papananos [119]. He implemented a weighted averaging algorithm, which is a series of multiply and accumulate operations on images using analog circuitry. Since then there have been various attempts to develop analog image processing chips.

Wyatt et al. initiated the MIT vision chip project where the aim was to develop a single chip with the sensor and processing circuitry integrated to perform early vision tasks [49]. This was followed by considerable research on implementing integrated image sensors with analog image processing capabilities built into them[18][20]. Some of these early vision chips did not have programmability built into them and they had to be used only for specific image processing tasks. The advent of Cellular Neural Network (CNN) based image processing architectures gave way to a slew of new programmable vision chips [73][79][106]. However, these CNN based vision chips are not easy to program and the performance of these processors is very susceptible to process and environment variation.

Neuromorphic computing, a concept developed during the 1980s has become a very popular tool for solving problems in machine vision applications, especially motion estimation. Neuromorphic computing uses digital, analog and mixed-signal circuits to mimic the neuro-biological architectures in the nervous system. In [29] the authors present such a neuromorphic chip and demonstrate decision making and bistable perception. In [36] the authors present a neuromorphic optical flow based reconfigurable hardware with properties of cortical motion pathway. The architecture is designed and tested on an Altera FPGA. In [37] the same authors present the implementation of the neuromorphic architecture on a Virtex FPGA. Many attempts have been made in literature to mimic the biological vision system to process images. In [93] the author presents a neuromorphic motion estimation model of the vision system in a fly which has a one-dimensional array of motion detectors, whose outputs are aggregated using a non-linear function. In [101] and [54] the authors develop a mixed signal neuromorphic chip to replicate the response of cells in the visual cortex and calculate a disparity map for natural scenes in real time. These systems required the use of FPGAs. While the performance of these architectures is good, the biggest drawback of implementing these architectures on an FPGA is the high power

consumption and large size. There have been various analog focal plane array implementations for optical flow based motion estimation. These compute the flow at a per-pixel level. As an example, the authors in [3] present such a design. One of the drawbacks of such an implementation is the high complexity of the circuit design at a pixel level and the low fill factor of the pixels in the image sensor. In [80] the authors present a mixed signal neuromorphic chip for address event based image processing and in [34] the authors present a multi-chip neuromorphic system where address event representation (AER) is used for inter-chip communication. Here the neuromorphic chips operate on data represented in the AER format and not on a frame based format. This requires the use of dynamic CMOS image sensors with pixels that respond to events. Some of the drawbacks of these sensors are that they are highly susceptible to transistor mismatch, they have a large pixel size and very low fill factors. With the end of CMOS scaling approaching soon, technologies beyond CMOS are already being considered and memristors are one of the most promising of these technologies. In [118] the authors present a neuromorphic chip based on CMOS-memristor technology. While they show that the functionality of the memristor based neuromorphic chip for pattern recognition is good, the memristor technology needs to achieve a low resistance state of the order of mega ohms (MΩ) to implement a practical low power brain-inspired computing chip. Some other challenges with a memristor based computing chip are handling the sneak path current issues and device variability.

There have been various other implementations of analog processors for vision-based systems but none of them, to our knowledge, have implemented an analog prototype board using COTS components for machine vision and image processing applications.

3.2. Reasons for developing the analog processing board

Field Programmable Gate Arrays have transitioned from simple glue logic to reprogrammable silicon chips that can replace digital application specific integrated circuits. Developing an analog version of an FPGA that is capable of performing all analog circuit functions is a challenging task. The major reason for this being that, not all complex analog circuit functions can be represented as a combination of simpler analog circuits. There have been implementations of reconfigurable analog chips which are called Field Programmable Analog Arrays (FPAAs) [24][105]. The advantage of FPAAs is that they remove the fabrication step from the iterative process and hence dramatically reduce the time for the design cycle. However, the FPAAs have their own drawbacks. A design implemented on an FPAA will have higher parasitics as well as increased die area. As a result of this, the design possesses inefficiencies like low bandwidth and increased power consumption. But, since the design of analog integrated circuits take time, the FPAAs can be a good trade-off.

The requirement in this research work however is slightly different. In this work, we are not trying to build a reconfigurable analog chip. The aim here was to build a rapid prototype of an analog processor using COTS components for the specific application of machine vision and image processing. Most machine vision and image processing algorithms make extensive use of computationally intensive, time consuming and iterative mathematical operations like convolutions and correlations which can be implemented easily in the analog domain. Also, using commercially available components has a lot of advantages. They are an excellent tool to demonstrate functionality. There is significant savings in terms of time and cost, as designing and testing custom parts is generally very time consuming and expensive. The off-the-shelf components are generally readily available which again saves time. They are also easy to modify

to fit to custom design requirements and they offer a lot of flexibility in case design changes or revisions have to be made.

3.3. Image Processing in the Analog Domain

Most of the machine vision algorithms perform low-level and mid-level image processing tasks which involve spatial filtering of images. Here a spatial filter mask or kernel performs a pre-defined operation on a neighborhood of pixel intensity values to create a new pixel-intensity value with the coordinates equal to the coordinates of the center of the neighborhood. For linear spatial filtering the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask [82]. In general, linear spatial filtering of an image *i* of size M x N with a filter kernel *k* of size m x n is given by:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k(s,t) i(x+s, y+t) \tag{18}$$

where m = 2a+1 and n=2b+1.

To generate a complete filtered image equation (18) is applied for $x = 0,1, \ldots . M - 1 \; and \; y = 0,1, \ldots . . N - 1$.

Spatial filtering of images can be performed using either of the two basic mathematical operations, correlation or convolution. In correlation, the filter mask is moved over the entire image and at each location the sum of products is computed. The mechanics of convolution are the same, but the filter is first rotated 180^{o}. Using convolution or correlation to perform spatial filtering of images is a matter of preference [82]. Grayscale images can be processed directly using these techniques. Full color image processing techniques are used widely in a broad range of applications. Two kinds of color image processing techniques are generally used. In the first

technique, each individual component of the color image is processed and then a composite image is formed and in the second technique, all color components are dealt with simultaneously.

The spatial filtering of images can be considered as a 2D filtering operation as shown in the equation (18) i.e. the multiplication operation is followed by two summation operations. When the 2D spatial filtering is implemented in the analog domain, one of the methods of implementing it is to perform continuous filtering operation in one dimension and discrete operation in the other dimension. Even though the second dimension is discrete, the operation is still performed in the analog domain. The 2D analog image filtering operation is given by the equation:

$$g(x,y) = \sum_{s=-a}^{a} \int_{t=-b}^{b} k(s,t)i(x+s,y+t)dt \qquad (19)$$

The discreteness in the second dimension is achieved by multiple analog channels where each analog channel performs a continuous time operation in the analog domain. The number of analog channels depends on the size of the filter kernel. Each row of the image intensity value and the filter kernel value is converted into an analog signal and processed independently by each analog channel i.e. these values are multiplied and then integrated. This accomplishes the computation in the first dimension. The outputs of each of these analog channels are summed in the analog domain and this accomplishes computation in the second dimension. Another important point to be considered while performing convolution or correlation is that of shifting the kernel values. The shifts in the kernel values are handled in the data readout for the analog prototype board. This is done by rearranging the image intensity values in a specific order in memory so that they represent a shift when they are read out.

3.4. Hardware Architecture

The analog processing board prototype has been developed using a variety of commercially available off-the-shelf components. The functionality of the prototype board is to implement mathematical operations such as convolution and correlation. The number of analog channels that are required depends on the size of the filter kernel being used. In this work, we have considered three analog channels. This means that the prototype can be used to demonstrate spatial filtering of images with kernels of size 3x3, which is the minimum filter kernel size used. Figure 19 shows the high-level block diagram of the analog processing board prototype with three channels. Consider a 3x3 image and a 3x3 kernel. The first row of the image and kernel is input into the first multiplier section (Input 1 and Kernel 1), second row into the second multiplier section (Input 2 and Kernel 2) and the third row into the third multiplier section (Input 3 and Kernel 3) as shown in the figure. The kernel and image values are multiplied in the multiplier

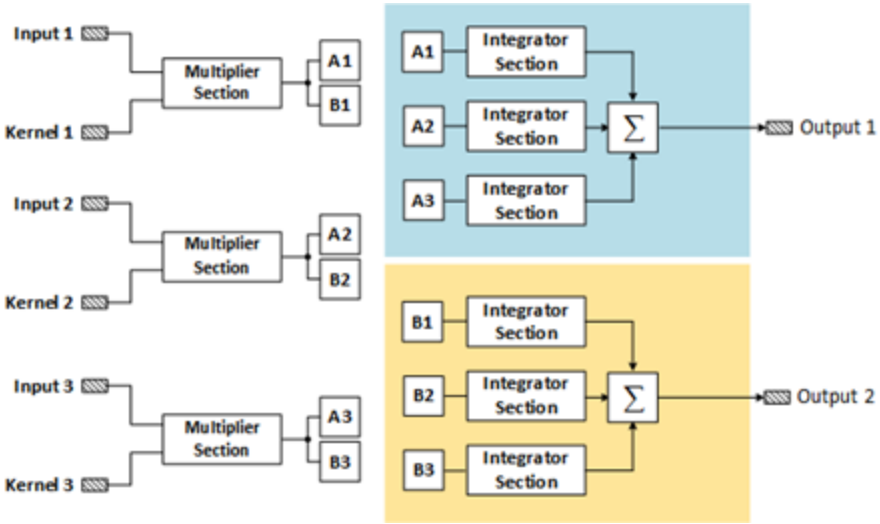


Figure 19: High-Level block diagram of the analog processing board prototype

section and the outputs (and their copies) represented by A1(B1), A2(B2) and A3(B3) are sent to the integrator section where they are integrated. This completes the first dimension of

computation as shown in equation (19). To complete the image filtering operation, the integrator outputs must be summed as shown in equation (19). This is accomplished through a voltage summer as shown in Figure 19. The output of the voltage summer is a filtered pixel. There are two integrator sections as can be seen from the figure. This is for an interleaved operation which is explained in greater detail in section 3.5.

The architecture of the prototype board can be divided into two major sections, the multiplier section and the integrator section.

3.4.1. The Multiplier Section

The multiplier section of the prototype board is responsible for receiving analog inputs, performing signal conditioning and multiplication. The analog prototype board has three channels in the multiplier section. These three channels can be used to perform spatial filtering using a 3x3 kernel or they can also be used for any other 3-channel image processing task. The block diagram of one channel of the multiplier section is shown in Figure 20. The components have been numbered to identify them. An image processing algorithm using an N x N filter kernel would have N such multiplier sections.

3.4.1.1. SMA Connectors

The analog values of the input image intensities and the filter kernels are passed through standard SMA female connectors (‘Amphenol 132417’) which are indicated by components 1 and 2 in the figure. There are two SMA input connectors for each multiplier section.

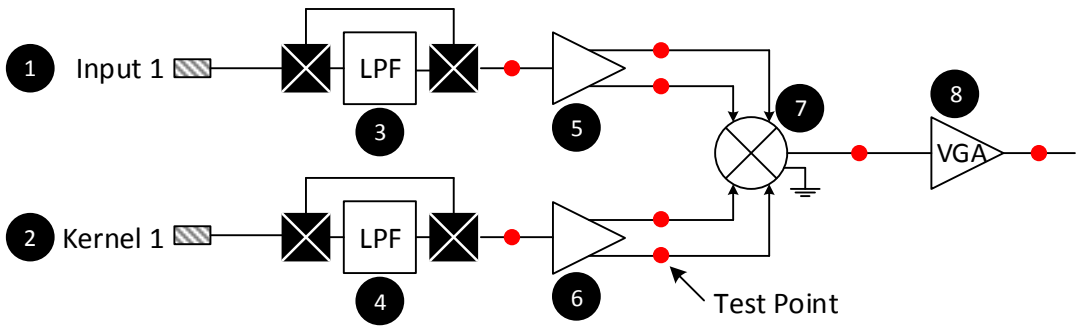


Figure 20: Block diagram of one channel of the multiplier section of the analog prototype board

3.4.1.2. Low Pass Filters

The analog image input intensities and filter kernel values can then be passed through a low pass filter. This is indicated by components 3 and 4 in the figure. The low pass filter selected here is RLP-70+ which is a component from "Minicircuit". This is a passive low pass filter with a passband from DC to 70MHz with an insertion loss of less than 2dB. This implies that the highest frequency of operation of the prototype board can go up to 70MHz. The low pass filter will be used to remove any high frequency noise and out-of-band signals on the analog intensity or filter kernel signals. A jumper around the low pass filter can be used if the filter must be bypassed. If the frequency of operation of the analog board must be increased to 'f' MHz, the pixels and kernel values will be read out at this frequency. This would mean the pass band of the low pass filter selected should be from DC to 'f' MHz. The connections for the low pass filter are obtained from the manufacturer's datasheet [65]. The filter has one input and one output, and all other pins are grounded.

The important circuit parameter here is the passband frequency. If the passband frequency is low, then the required signals may not go through and if the passband frequency is

too high then this will result in unwanted signals going through which may contribute to the noise at the output.

3.4.1.3. Single-Ended to Differential Amplifier

The multiplier that has been selected for the prototype board expects a differential input. The two analog input signals are single-ended. Hence, we need a single-ended to differential converter (components 5 and 6). The single-ended to differential converter has been designed using an op-amp. The op-amp selected for this operation is LMH6552 from Texas Instruments. This is a fully differential amplifier with a 3dB bandwidth of 1.5GHz. Figure 21 shows the circuit diagram of a single-ended to differential amplifier built using a fully differential op-amp. The design equations for the single-ended to a differential op-amp are given below [102]:

$$A_v = \left(\frac{2(1-\beta_1)}{\beta_1+\beta_2}\right) \qquad \beta_1 = \left(\frac{R_G}{R_G+R_F}\right) \quad \beta_2 = \left(\frac{R_G+R_M}{R_G+R_M+R_F}\right) \tag{20}$$

$$R_{IN} = \left(\frac{2R_G+R_M(1-\beta_2)}{1+\beta_2}\right) \tag{21}$$

Here A_v is the required voltage gain of the single-ended to differential amplifier, R_S is the source resistance, and R_{IN} is the input impedance of the amplifier. The design requirement is that $R_T||R_{IN} = R_S$ and $R_M = R_T||R_S$. The source resistance is generally 50 Ω. R_F is the feedback resistance of the amplifier and the values of both feedback resistors are kept the same. The resistors connected to the inverting and the non-inverting input terminals, R_G are also kept the same. This means that the ratio of R_F/R_G sets the gain of the amplifier output. From the manufacturer's datasheet [102] we obtain information that, for optimum operation of the design, the value of the feedback resistor must be between 270 Ω and 390 Ω. Once this value is fixed we can calculate

the value of R_G depending on the gain requirement. This will allow us to then calculate β_1 and β_2 from which we can calculate R_M.

Solving these design equations using the above method gives the values of the all the required resistors. The resistance values obtained for this design are as follows:

R_s = 50Ω, R_T = 59Ω, R_G = 255Ω, R_M = 27Ω and R_F = 275Ω.

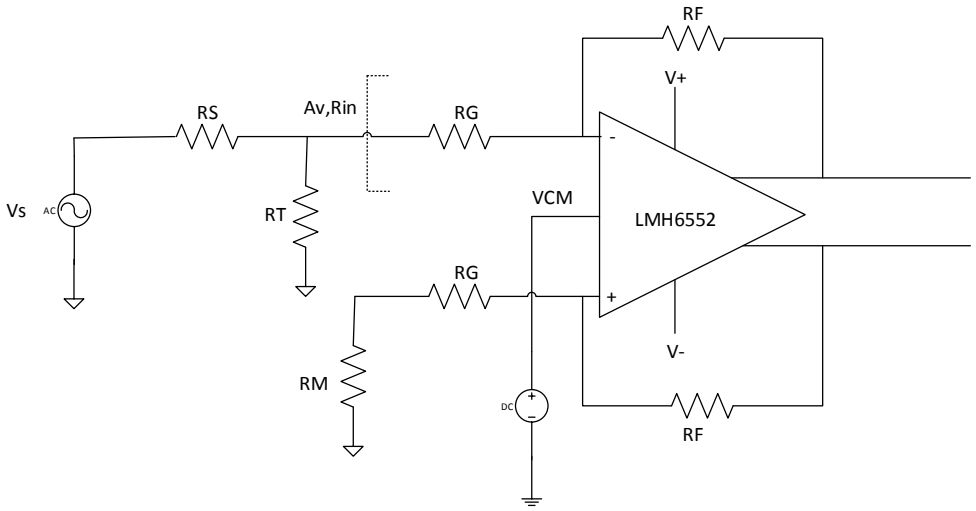


Figure 21: Circuit diagram of a single-ended to differential amplifier using a fully differential op - amp

The important parameters for this component are the resistor values. These resistor values set the gain at the output of the amplifier and should be as close as possible to the theoretical values calculated from equations (20)and (21). Hence, we have used resistors with 0.1% tolerance. If these resistor values deviate more than this, the gain of the single-ended to differential amplifier will vary and this will result in incorrect values at the input of the multiplier.

3.4.1.4. Analog Multiplier

The analog multiplier (component 7) is one of the most critical components in the design of the prototype board. The multiplier is used to multiply the analog image intensity values with filter kernel values. This is the first step of the convolution process. Since the filter kernel values can be both positive and negative we needed an analog four quadrant multiplier. The input dynamic range was set to be 1V. We also wanted a multiplier with a very low noise figure so that the performance would not be affected by noise. The multiplier that had specifications closest to this was Analog Devices', ADL5391. This is a fully differential analog multiplier that has a multiplication bandwidth from DC-2GHz and an input dynamic range of +2V to -2V. The transfer function of the multiplier is given by:

$$V_W = \frac{\alpha(V_X \,.\, V_Y)}{1V} + V_Z \tag{22}$$

where, V_X and V_Y are the input analog voltages to the multiplier, α is the multiplier gain and is set to 1, and V_Z is the summing voltage input which is set to 0 in the prototype board design. The output of the ADL5391 is also differential. The performance of the multiplier was critical to the performance of the analog prototype board. Hence, the ADL5391 multiplier functionality was first verified against the results provided in the datasheet using an evaluation board. The connections for the multiplier are obtained from the manufacturer's datasheet [9]. For a general design with N analog channels, N such multipliers can be used.

The important parameters that must be set for the multiplier are the gain and summing voltage. The gain α is always set to 1. The summing input V_Z must be set to 0. It is important to ensure that the gain of the multiplier is fixed and known. If the gain of the multiplier deviates and becomes unknown, the multiplier outputs will be erroneous.

3.4.1.5. Variable Gain Amplifier (VGA)

The output of the multiplier is passed through a VGA (Component 8). The VGA allows the gain of the multiplied output to be changed i.e. amplified or attenuated. Even though the previous components have been set to have no gain, there may be minor variations in these components that make the gain at the multiplier output non-zero. The gain setting of the VGA here was tuned such that the output of the VGA had no gain with respect to the input at the SMA connectors. The VGA selected for this purpose is AD8336 from Analog Devices. The connections for this component were obtained from the manufacturer's datasheet [7]. It is a low-noise, single-ended general purpose VGA with an absolute gain variation of -26dB to 34dB and a bandwidth of 80MHz. The gain variation of the VGA chosen in this design will be sufficient for most of the applications and other general designs. However, if the frequency of operation of the prototype board must be increased, a VGA with a higher bandwidth capability must be chosen.

The parameter that is controlled for the VGA is the voltage applied to set the gain. It is a variable DC voltage that can be applied to the gain control pin through a potentiometer.

3.4.1.6. Test Points

The aim of developing a prototype board was to verify the functionality of the board and the components. Hence, test points have been placed at the input and output of all the components. This is indicated by red dots in Figure 20.

3.4.2. The Integrator Section

The integrator section of the analog prototype board performs integration of the multiplied signals which accomplishes the 1D convolution or correlation. It is also responsible for the computation in the second dimension. This is accomplished by adding the 1D outputs from all

the three channels using a summing circuit. Figure 22 shows the block diagram of the integrator section. The components have been numbered to identify them. In the figure, A1, A2 and A3 are the analog outputs from the three channels of the multiplier section.

3.4.3.1. Unity Gain Buffer

The outputs of the VGA from each analog channel are input into to a unity gain buffer indicated by components 9,10 and 11. This ensures that the VGA output sees a high impedance stage. The unity gain buffer amplifier was designed using an op-amp. The component selected for

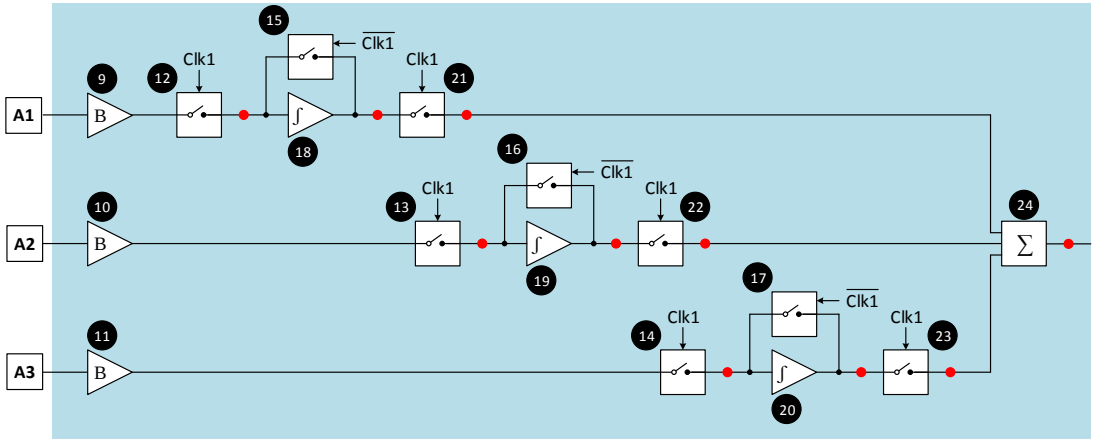


Figure 22: Block diagram of the first integrator section

this is ADA4817, an op-amp from Analog Devices. This is a low-noise, unity-gain stable, high speed voltage feedback amplifiers with FET input stages. The unity gain buffer is a simple non-inverting amplifier with a voltage gain, A_V of 1. This is designed by making the feedback resistor small and the resistor connected to the inverting input very large. We use the manufacturer's datasheet for reference design [6]. The same unity gain buffer can be used for any general design with N analog channels. The parameters for the unity gain buffer are the values of the feedback resistor and the resistor connected to the inverting input terminal. These values need not be very accurate. It must be ensured that the ratio of the feedback resistor value to the resistor connected to the inverting

input terminal is very small. If these values are off, the gain will not be unity which will result in errors at the output.

3.4.3.2. Integrator

The basic op-amp based integrator circuit consists of an op-amp with a capacitor between the output and the inverting terminal and a resistor at the inverting input terminal as shown in Figure 23. The integrator requires an op-amp that has high slew rate and very low input bias currents. The op-amp that was selected was ADA4817 from Analog Devices. The output voltage of the integrator is given by $V_{out} = -\int_0^t \frac{V_{in}}{R_1 C_f} dt$.

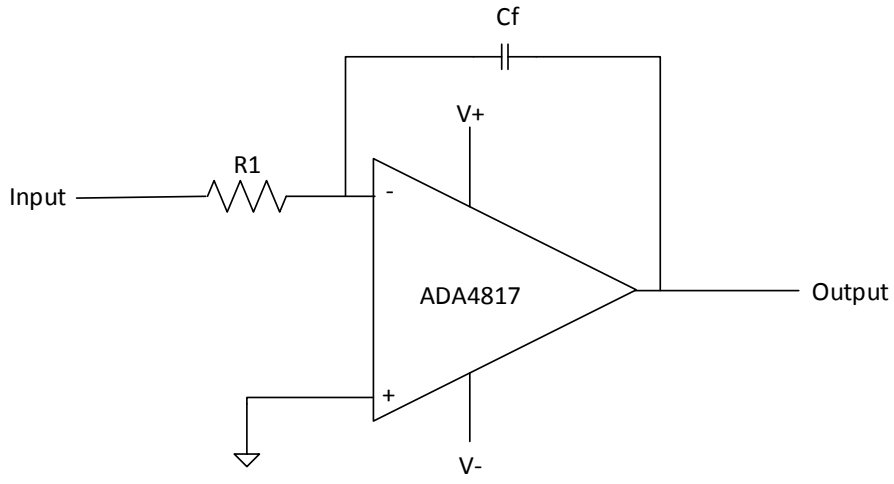


Figure 23: Circuit diagram of the op-amp Integrator

The parameters for the integrator design are the resistor and feedback capacitor values. The resistor value is generally fixed and should have a low tolerance. Here we use resistors with 0.1% tolerance. The capacitor value should also be accurate as this determines the integration frequency. The combination of resistor and capacitor also sets the gain of the integrator. If the values of the resistor and capacitor deviate, the gain of the integrated output will not be unity and the integration frequency will also vary, which will cause errors in convolution.

3.4.3.3. Reset Switches

The integrator is an analog continuous time circuit that integrates the input voltage till the output voltage reaches saturation. The prototype board is being used to perform spatial filtering of images using 3x3 kernels. Hence, the output of the integrator must be sampled after every 3 pixels and then a new integration cycle should begin. This is accomplished by using an RF switch in the feedback loop indicated by components 15,16 and 17 in parallel to the feedback capacitor C_f of the integrator. This switch is used to discharge the capacitor after three pixels have been integrated. A clock distribution chip on the prototype board is responsible for turning the switch on and off. The switch selected for this is ADG601/602 from Analog Devices. The connections for this component are obtained from the manufacturer's datasheet [8]. For a general design, if the readout frequency of the image intensity and kernel values is '*f*' MHz and the size of the kernel is *N* x *N* then the frequency at which the feedback capacitor C_f of the integrator is cleared must be *f/N* MHz.

3.4.3.4. Voltage Summer

The 1D convolved outputs from the three integrators must be summed to perform the 2D convolution. This is accomplished using an op-amp based three input inverting voltage summer circuit (Component 24). We use the same op-amp, the ADA4817 for the voltage summing circuit. We use the manufacturer's datasheet for the reference design [6]. For a general design that has N analog channels, an N-input voltage summing amplifier can be designed. When N becomes very large it is better to have a bank of voltage summers with lesser inputs.

The important parameter for the voltage summer are the resistor values connected to the inverting and non-inverting terminals of the op-amp. The resistors connected to the inverting terminal must be of the same value and must have very low tolerance (0.01%). The ratio of the

feedback resistor and the resistor connected to the non-inverting terminal sets the gain of the summer. The ratio should be very small to get unity gain. If the resistance values are not accurate, the result of the summing operation will be in error causing an error in the 2D convolution operation.

3.5. Interleaved Architecture

The prototype board has 3 analog channels in the multiplier section and three corresponding channels in the integrator section. This architecture can be used to perform spatial filtering using a 3x3 kernel. However, the integrator must be reset after every third pixel and for this the capacitor should be discharged completely. This process is not instantaneous and may take one or more clock cycles. This implies that the input data cannot be processed until the integrator has been completely reset. In order to circumvent this problem and maintain a continuous output, a second integrator section is used as shown in Figure 24. The timing diagram for the interleaved operation is shown in Figure 25. There are two clocks that are used which are inverted versions of each other. During the first three clock periods t0, t1 and t2, Clock1 (Clk1) is high. This closes the switches 12, 13, 14 ,21, 22 and 23 of the first integrator section shown in Figure 22. The outputs from the three multiplier analog channels denoted as A1, A2 and A3 are passed to the first integrator section. At the same time, Clock 2 (Clk2) being the inverted version of Clock 1 keeps the switches in the second integrator section open and hence no signal is passed here. At the end of t2, the integrators in the first section have finished integrating and must be reset. These switches are controlled by $\overline{\mathrm{Clk1}}$. During reset, Clk1 goes low and $\overline{\mathrm{Clk1}}$ goes high which opens the switches 12, 13, 14, 21 ,22 and 23 and closes the switches 15,16 and 17. This resets the integrators in section 1. At the same time Clk2 goes high for the next three clock periods t3, t4 and t5. This closes the switches 12, 13, 14 ,21, 22 and 23 of the second integrator section shown

in Figure 24 and passes the signals from the three analog channels denoted as B1, B2 and B3 into the second integrator section. Now the second integrator section performs the integration operation. This process is continuously repeated. The interleaved architecture enables us to obtain a continuous output and one image pixel is filtered out every three clock periods.

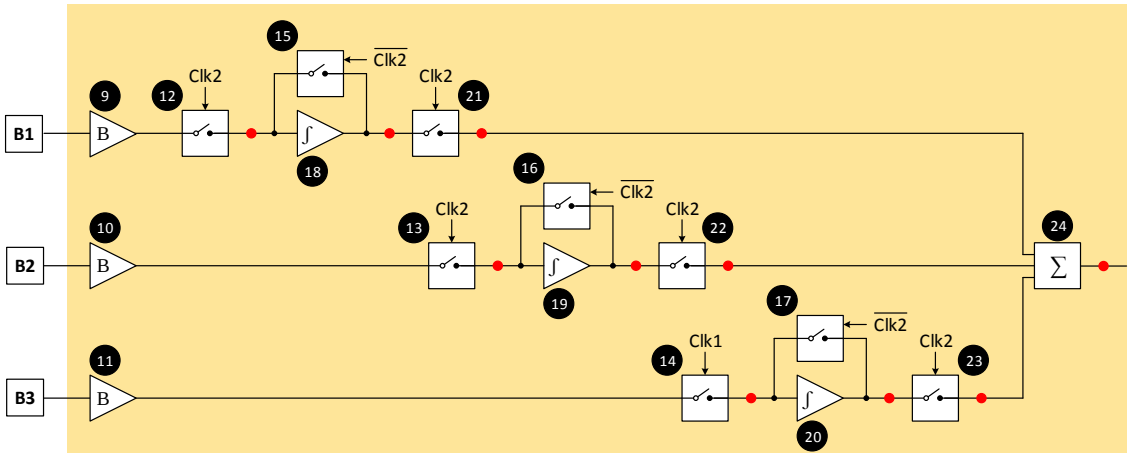


Figure 24: Block diagram of the second integrator section

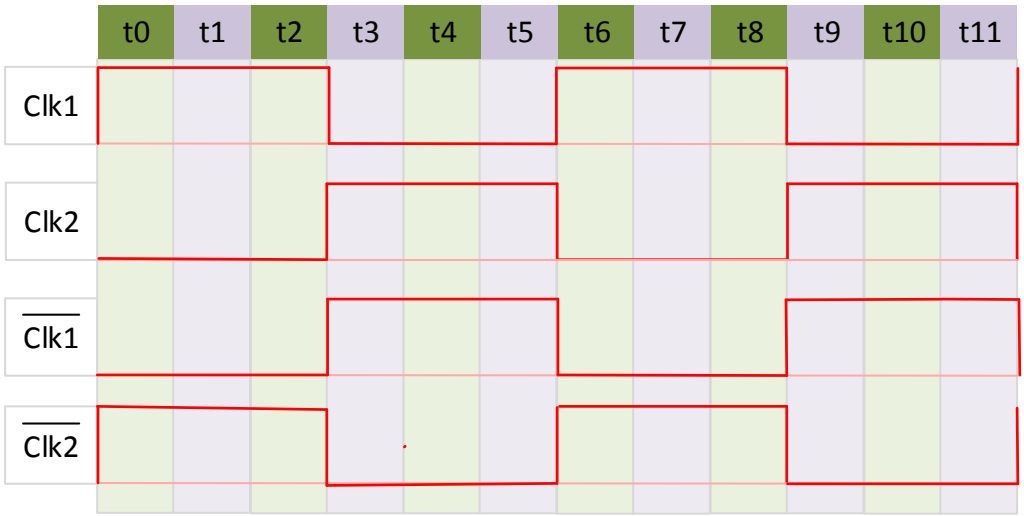


Figure 25: Timing Diagram for Interleaved Operation

3.6. Simulation of Analog Prototype Board

Before developing the prototype board, it is important to test the functionality of the board and its major components through simulation. The simulation tool used in this work is Advanced Design System (ADS). Most of the components selected for the design of the prototype have SPICE models available and this was used in the simulation. There are some components for which the SPICE models were not available. For these components, a behavioral model was created, and this was used in simulation. The first step in the simulation was to verify the performance of each component individually. These results were then compared to the results in the datasheets of the components. Once satisfactory results were obtained, the simulation was then carried out for one channel of the analog prototype board.

3.6.1. Schematic for simulation of one analog channel

Figure 26 shows the schematic of the multiplier section of one channel of the analog prototype board. In this schematic, the low pass filter has been by-passed and the analog signals are directly sent to the single-ended to differential converter. The single-ended to differential converter has been designed using the SPICE model for LMH6552. The resistor values are obtained using the design equations as shown in section 3.4.1.3. The differential outputs of the single-ended to differential converter are the inputs to the multiplier. The multiplier ADL5391 does not have a SPICE model associated with it. To simulate the performance of the multiplier, a behavioral model was developed in ADS using standard components. We also use a behavioral model for the Variable Gain Amplifier (VGA). This is just an amplifier whose gain can be controlled with the application of an external voltage. The output of the VGA is DC blocked using a capacitor. This ensures that the integrator does not saturate because of DC voltages.

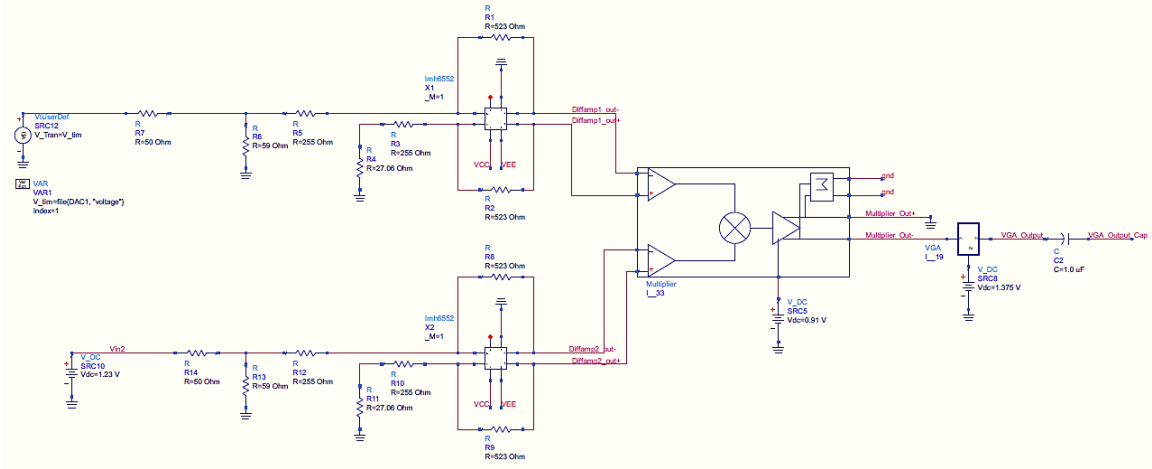


Figure 26: Functional Schematic of the Multiplier Section

Figure 27 shows the schematic of the integrator section. The output of the VGA goes to the input of a unity gain buffer. The buffer is designed with an op-amp, ADA4817 for which a SPICE model is available. The output of the buffer is fed into the integrator. The integrator has been designed for integrating signals at a frequency of 1 MHz and at this frequency the gain should be unity. From the design equations in section 3.4.3.2. we can calculate the feedback capacitance of the integrator to be 1nF.

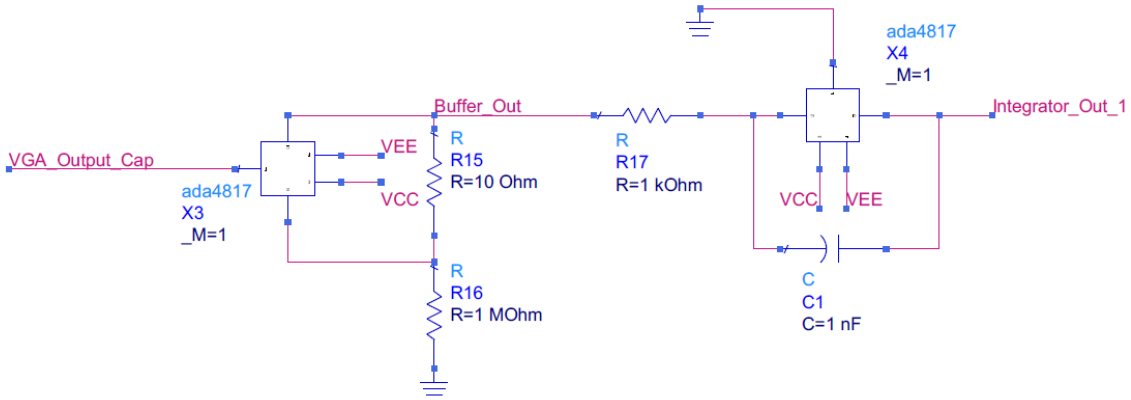


Figure 27: Functional Schematic of the Integrator Section

3.7. Simulation Procedure and Results

The analog prototype board has been designed to perform image filtering operations. In order to demonstrate the performance of the prototype board we choose two important and frequently used image processing algorithms. The first one is a RGB to YUV color transformation which is a color image processing algorithm and the second is an edge detection algorithm which can operate on grayscale images. The performance of the analog prototype board is compared to the performance of a digital system by implementing the same algorithms in MATLAB.

3.7.1. Simulation Procedure

To perform analog image processing, the digital image intensity values and kernel values are converted into analog voltage representations in MATLAB. The input images have 8-bit intensities, so the image intensities can have a maximum value of 255 and a minimum value of 0. The dynamic range of the analog voltage representation varies from 1V to 0.028 V, i.e., an intensity value of 255 is represented as 1V and an intensity value of 0 is represented as 0.028 V in the analog domain. This gives a voltage step of 3.81mV for each intensity value. The kernel values are directly converted into analog voltages i.e. the analog voltage value is the same as the digital value. The analog image and kernel values are input into the analog channel which performs the convolution operation. The output of the integrator is sampled and stored. The second dimension of summation is also done in the analog domain by a voltage summer with the stored integrator values as inputs. The output of the summer is sampled and stored. These analog voltages are converted to 8-bit digital values using MATLAB and the resultant outputs are displayed.

3.7.1.1. Performance Metrics

The results of image enhancement or processing can be very subjective and hence it is necessary to establish quantitative measures to compare the performance of image processing techniques. Peak-Signal-to-Noise Ratio (PSNR) is one such metric that is used frequently to measure the quality of output images. PSNR is defined as the ratio of the maximum possible value (power) of the signal to the power of noise that distorts the signal.

$$PSNR = 10log\left(\frac{MAX_I^2}{MSE}\right) \tag{23}$$

where MAX_I is the maximum possible pixel value for the image and *MSE* is the mean square error. *MSE* allows us to compare the true pixel value of the original image to the degraded image [86]. The higher the value of PSNR the better is the performance. Here we can consider the MATLAB outputs as the true value images and compare the performance of analog image processing by measuring the PSNR.

Another quantity that is used for measuring the similarity between two image windows *X* and *Y* is the Structural Similarity Index (SSIM). SSIM is a perception based model that considers image degradation as a perceived change in structural information while incorporating perceptual phenomena. Structural information is the idea that pixels have a strong interdependence especially when they are spatially close. It is computed as:

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{24}$$

In this equation μ_x and μ_y are the average of *X* and *Y*. σ_x^2 and σ_y^2 are variances of *X* and *Y*. σ_{xy} is the covariance of *X* and *Y*. $C_1 = (\kappa_1 L)^2$ and $C_2 = (\kappa_2 L)^2$, where *L* is the dynamic range of pixel values and κ_1=0.01, κ_2=0.03 [27].

3.7.2. RGB to YUV color transformation

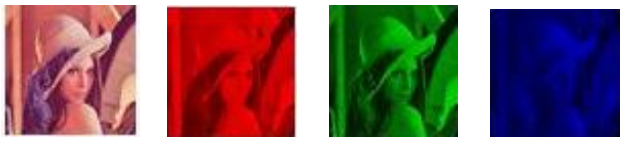
YUV is a color space that is used as a part of an image pipeline. It encodes a color image by taking human perception into account. This allows for reduced bandwidths for chrominance components thereby enabling transmission errors or compression artifacts to be more efficiently masked than a direct RGB transmission. The color transformation from RGB to YUV can be performed using the equations below. Here Y is the luminance component, U and V are the chrominance components [26].

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = -0.147R - 0.289G + 0.436B$$

$$V = 0.615R - 0.515G - 0.1B$$

Figure 28 shows a 64x64 pixel 8-bit input color image and its corresponding red, green and blue components. The digital image intensity values and kernel values are converted into analog voltage representations. For RGB to YUV conversion each individual color component image can be processed independently and then added together to get a color transformation.



(a) (b) (c) (d)

Figure 28: (a) Original Lena Image (b) Red Component of Lena Image (input) (c) Green Component of Lena Image (input) (d) Blue Component of Lena Image (input)

Figure 29 (a), (b) and (c) show the results of performing a RGB to YUV conversion on 64x64 Lena image in the analog domain. When an image is processed or enhanced for visual interpretation the viewer is the ultimate judge of the performance. Hence it is important to

compare the images qualitatively. The YUV outputs from the analog processor simulations are compared to the YUV outputs from MATLAB which are shown in Figure 29 (d), (e) and (f).

The PSNR for the analog Y component is 53.71dB, analog U component is 55.26dB and analog V component is 50.9dB. These PSNRs were measured by taking the MATLAB simulated outputs as true images. The PSNR value of the output image of a standard image compression algorithm like Joint Photographic Experts Group 2000 (JPEG2000) can range from 30dB to 50dB [30] and the quality of this compressed image is accepted worldwide. In comparison, the PSNRs for all the three outputs are above 50dB and hence, we can conclude that the performance of the analog prototype is good.

The SSIM values were calculated between the analog and digital values for each of the Y, U and V channels and were found to be 0.9983, 0.9783 and 0.9782 respectively. These high values of the SSIM show that the outputs of the analog image processing operation are very close to digital processing results.

Figure 29: (a), (b) and (c) show the results of analog RGB to YUV conversion in ADS. (d), (e) and (f) show the results of RGB to YUV conversion performed in MATLAB

3.7.3. Edge Detection using Prewitt Kernels

Edge detection operation identifies points in an image at which the brightness changes sharply or has discontinuities. The points at which there are discontinuities are typically organized

into curved line segments called edges. Edge detection is a very fundamental tool in machine vision algorithms particularly in object detection and feature extraction.

A Prewitt kernel is used as the operator for finding edges here. It is a discrete differentiation operator computing the approximation to the gradient of the image intensity function. Prewitt operator gives the rate of change in a particular direction and also information on how the edge is likely to be oriented [82]. The kernels used for edge detection are given by,

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} and\ K_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Convolving the image with these kernels gives the edges in the x and y direction respectively. Figure 30 (a) shows the 64x64 pixel grayscale image input. The image intensity values and kernel values are converted into analog voltage representations and are input to the analog channel in ADS. Figure 30 (b) and (c) show the analog output obtained from ADS for horizontal and vertical edge detection using the Prewitt kernel. Figure 30 (d) and (e) show the output obtained from MATLAB for horizontal and vertical edge detection using the Prewitt kernel. As can be seen from the figure the performance of analog image processing is very close to that of digital processing.

The PSNR was measured for both horizontal and vertical edge detection in the analog domain by taking MATLAB outputs as true images and it was found to be 50.7dB and 50.56dB respectively. The SSIM values for horizontal and vertical edge detection in the analog domain are 0.9782 and 0.9676 respectively. These results show that the performance of the analog processing board is at par with a digital implementation.

(a) (b) (c) (d) (e)

Figure 30: (a) 64x64 pixel grayscale input image (b) Analog output of edge detection in X-direction (c) Analog output of edge detection in Y-direction (d) Digital output of edge detection in X-direction (e) Digital output of edge detection in Y-direction

3.8. Design of the analog processing board prototype

The analog processing board prototype is designed on a printed circuit board (PCB). The PCB schematic design of the analog processing board prototype was done in ADS. The components used on the analog prototype board have different power supply requirements. To address this, a single power supply is used along with buck converters (step-down converters) and voltage regulators. Clock distribution chips are used to control the switches responsible for the interleaving operation and clearing the integrators.

Figure 31 shows the layout of the analog processing board with three multiplier sections and two three-channel integrator sections along with the support circuitry. The analog processing board is built on a 6-layer FR4 board which is 93mils thick. The dimension of the board is 10.5” by 14”.

3.9. Measured Results

Figure 32 shows a fully populated analog processing board which was used for testing purposes and Figure 33 shows the block diagram of the test setup used for testing the analog processing board. MATLAB is used to convert digital image and kernel values into analog voltage representations. The analog voltage representations are converted into analog signals using an arbitrary function generator. The analog signal inputs are processed using the analog processing board. The processed outputs of the board are displayed as waveforms on the oscilloscope. The scope output voltages are sampled and stored. MATLAB is used to convert these voltages back to image intensity values. Figure 34 shows the actual test setup used for measurements.

As mentioned before, two image processing algorithms are used to test the performance of the analog processing board prototype. One is RGB to YUV color transformation and the other is edge detection. As shown in section 3.6, through simulation in ADS, we were able demonstrate that the analog processing board prototype can achieve a good performance, and this was shown on complex image inputs. For actual measurements, due to limitations of the test equipment and setup, we use images that are less complex to demonstrate the functionality of the analog processing board prototype. However, we still use the same performance metrics to evaluate the measured results and compare it to digital simulations in MATLAB.

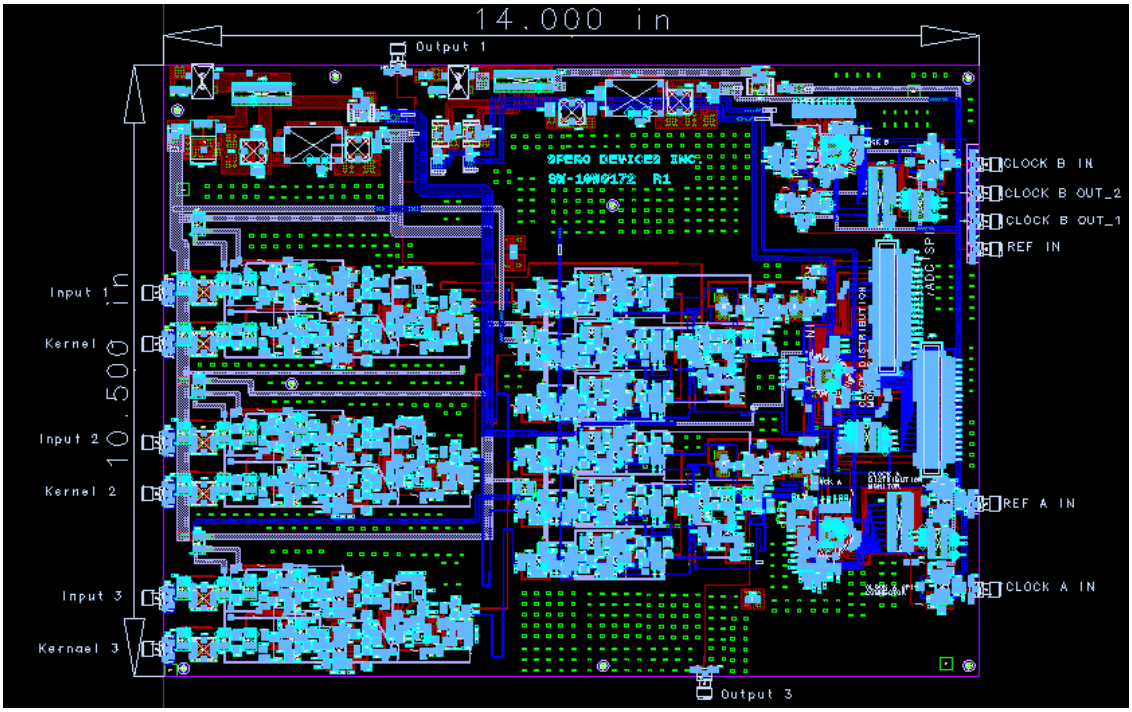


Figure 31: Layout of the analog processing board prototype

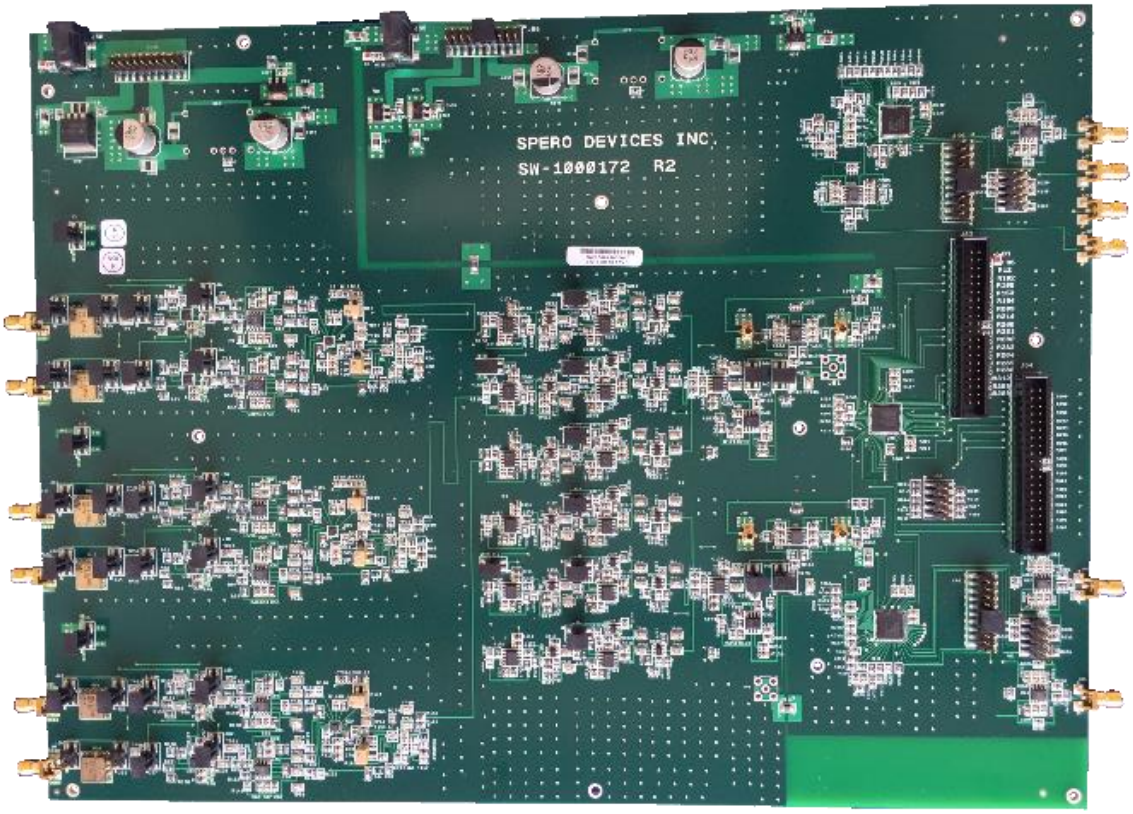


Figure 32: Fully populated analog processing board used for testing

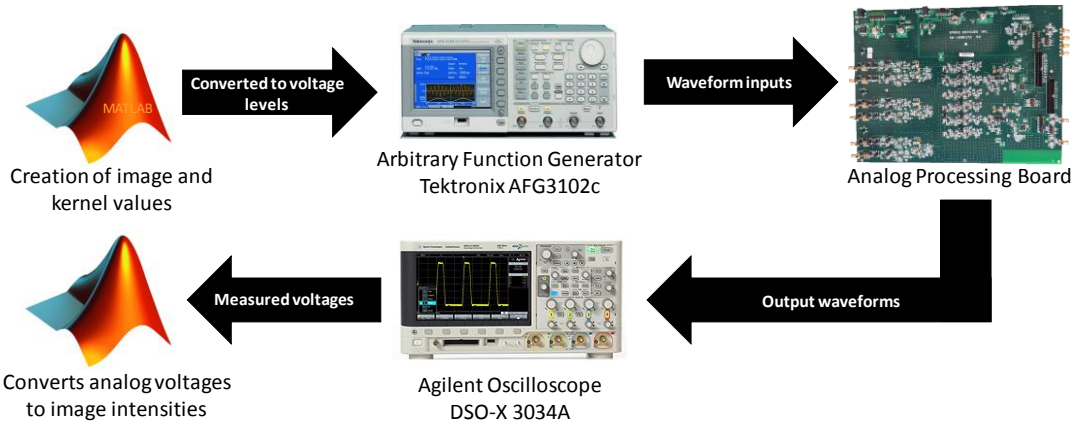


Figure 33: Block Diagram of the test setup for the analog processing board prototype

Figure 34: Test setup for testing the analog processing board prototype

3.9.1. RGB to YUV color transformation

Figure 35 (a) shows the input RGB image and its components used to perform RGB to YUV color transformation on the prototype board. Figure 35 (b) shows the expected digital output for a RGB to YUV color transformation generated in MATLAB. Figure 35 (c) shows the YUV image generated from measured outputs of the analog prototype board.

The PSNR for the analog, measured Y component is 54.64dB, measured U component is 57.89dB and measured V component is 58.16dB. These are high PSNRs when compared to a JPEG2000 image compression algorithm and they indicate a good performance of the analog processing board prototype and show that noise and analog computation errors can be tolerated in image processing applications.

The SSIM values were calculated between the images generated from measured voltage values and MATLAB generated digital images for each of the Y, U and V components and were found to be 1, 0.9999 and 1 respectively. We obtain very high values of the SSIM which shows

that the measured results from the analog board are very close to the MATLAB generated digital outputs.

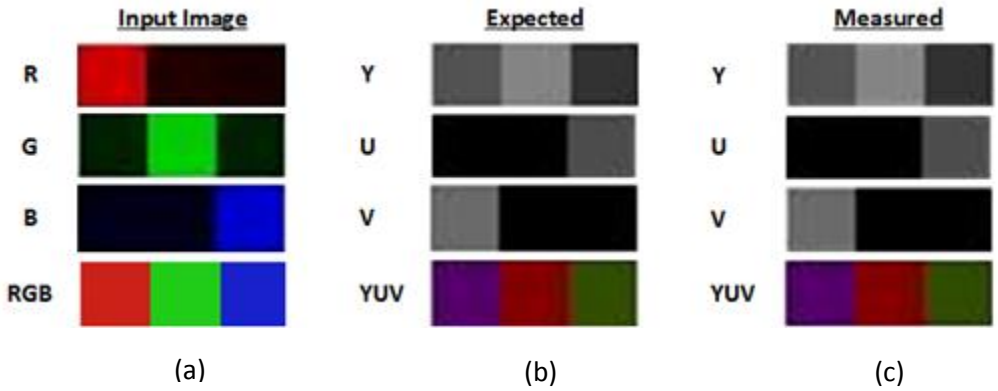


Figure 35: (a) Input RGB image with red, green and blue components (b) Expected YUV output and its components generated in MATLAB (c) Measured YUV outputs and components from the analog board

3.9.2. Edge Detection Using Prewitt Kernel

Figure 36 (a) shows the input binary image used to verify the performance of the analog processing board for edge detection using a Prewitt kernel. Figure 36 (b) shows the image generated from measured voltage values for horizontal edge detection using a Prewitt kernel. Figure 36 (c) shows the image generated from measured voltage values for vertical edge detection using Prewitt kernel.

The input image is binary i.e. it has only two intensity values, 0 or 1. These image intensities are represented in the analog domain as 28mV and 1V. The processed output image is also binary and will be represented by these two voltage levels which are then converted to image intensity values. Since the input and output images are binary, the outputs of the analog processing board are the same as that of a digital system (MATLAB generated output). This was verified by measuring the PSNR and SSIM values which were found to be infinity and one

respectively. This shows that the performance of the analog processing board prototype is extremely good for binary images.

(a) (b) (c)

Figure 36: (a) Input Binary Image (b) Measured results of horizontal edge detection using Prewitt kernel (c) Measured results of vertical edge detection using Prewitt kernel

3.9.3. Signal to Noise Ratio Measurements and Accuracy of the analog processor

One of the drawbacks of analog circuits is that they add noise, and this may affect the performance of the system. Hence, it becomes very important to measure the amount of noise added by the system. This can be done by measuring the input SNR and the output SNR. The Noise Figure in dB can then be calculated as the difference of the two SNRs. Figure 37 shows the test setup used to measure the SNR at various test points. Analog signals are input through an arbitrary function generator. We use a portable spectrum analyzer connected to a laptop to measure the signal and noise levels at each test point. From the measurements, it was found that the noise figure of the system is 25.1dB. The measured SNR for a test image was 53.5dB which agrees with the simulation results.

Friis's equation is used to measure the cascaded noise figure of a system and is given as:

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \cdots \quad (25)$$

where *F* is the noise factor. According to equation (25), the noise figure of a system is dominated by the components that appear before a gain stage. The components that appear

before a gain stage in our system are the single-ended to differential amplifier and the analog multiplier.

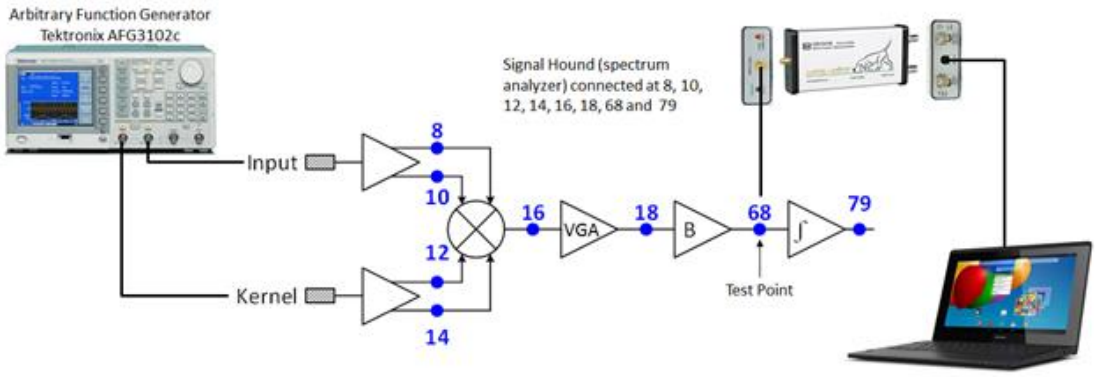


Figure 37: Test set up for measuring Signal to Noise Ratio at various test points

This means to further improve the performance of the board, the single-ended to differential amplifier stage and the analog multiplier must be designed with very low noise. As an example, we designed an analog multiplier in 65nm CMOS technology and simulated the noise performance of the multiplier. If the COTS multiplier is replaced by the new multiplier that was designed, the results show that the noise figure improves from 25.1dB to 9.3dB. This is a 15.8dB improvement in output SNR.

For an analog system, the output SNR translates to the bit accuracy that can be achieved, once the analog signal has been digitized. For an N-bit analog-to-digital converter the SNR is given as 6.02N + 1.76dB. For an 8-bit system the required SNR is 49.92dB. Both simulations and measured data show that the analog prototype board can achieve an output SNR of more than 50dB which translates to 8-bits of accuracy. This accuracy is lower than what the FPGAs and GPGPUs can offer, which is single or double precision floating point. However, 8-bits of accuracy is sufficient for most low-level and mid-level image processing applications. Also, the current industry trend is towards the development of custom hardware tensor processing units (TPUs) to

replace GPUs for emerging high-performance computing applications, particularly deep learning. For example, Google announced an 8-bit fixed point TPU for deep learning and Intel is developing a TPU with 16-bit fixed point architecture for deep learning applications.

3.10. Proposed Integrated Circuit Design of analog processor board

The analog processor prototype was implemented using COTS components. The aim here was to demonstrate the functionality of the analog processor and this was successfully demonstrated through experimental results. The use of COTS components has several advantages as mentioned before. However, it does not show speed and power improvement over existing digital systems and other analog processors proposed in literature. This can only be demonstrated through an integrated circuit implementation of the analog processor which we call the Analog Space-Time Convolution Processor (ASTCP).

3.10.1. Performance of the Integrated Analog Processor (ASTCP)

In this part of the dissertation, we make estimations of speed improvement, power consumption and area of the analog processor based on some of the integrated circuit components that we have previously designed and fabricated or based on integrated circuit components that are available in literature. Table 2 gives the list of major components, their power consumption values and the area occupied by each device. The frequency of operation of the analog circuit components was assumed to be 500MHz.

Figure 38 (a) shows the performance comparison of the ASTCP against the performance of FPGA (Xilinx Artix) [116], GPU (Nvidia's Pascal architecture) [71] and a state-of-the-art digital ASIC (Movidius Myriad 2) [68]. The performance comparison metric used is Giga Operation per Second per Watt (GOPS/W). As can be seen from the figure, the performance of the analog

processor when implemented as an integrated design is 100x better than FPGAs, 5x and 4x better than state-of-the-art GPU and digital ASIC respectively.

Table 2: Power Consumption of all the core components in one channel of the analog processor

Component	Power Consumption per device	Area of Each Device
D/A converters [112]	0.78mW	0.016 mm^2
Buffer Amplifier [5]	0.1mW	0.0052 mm^2
Multiplier [4]	0.2mW	3.68e-6 mm^2
Integrator [4]	0.193mW	0.0052 mm^2
Summer [4]	0.549mW	0.0052 mm^2
A/D converters [78]	0.82mW	0.047 mm^2

Figure 38 (b) shows the speed comparison for the same devices. Here the operation being considered is a simple image filtering on a 5-megapixel image using 2 different filter kernel sizes, 3x3 and 5x5. As can be seen, the analog processor can achieve extremely high frame rates which can be very critical for more complex machine vision applications like object detection and recognition.

(a) (b)

Figure 38: (a) Performance comparison of different devices in GOPS/W (b) Performance comparison of various devices in Frames per second for filtering a 5MP image with two kernels of sizes 3x3 and 5x5

Table 3 shows the performance comparison of the Analog Space-Time Convolution Processor (ASTCP) against other analog processing based hardware implementations for machine vision applications. Most of these implementations are CMOS based, but in [69] the authors have

built a processor by using a combination of CMOS and spintronic based devices. Each processor can handle a different frame size and frame rate and to compare the performance we consider a figure of merit (FOM1). As can be seen from the table the ASTCP presented in this work has the best FOM.

A good processor design should always achieve an optimum cost-performance tradeoff at a particular target performance. There are two parts to the cost of a chip. The first one is a fixed cost also known as the Non-Recurring Engineering (NRE) cost which accounts for the chip design, verification and mask generation. The second one is the recurring cost which is proportional to product volume and it includes the chip area, packaging and test. The NRE cost always keeps increasing and multiple design runs of a chip adds to this cost. In this work, we have addressed this issue by first designing a prototype board using COTS components. Once the design has been finalized, we can move into an integrated chip design phase, which will now be cheaper.

The recurring cost can be reduced by having a smaller chip area. The digital hardware accelerators generally occupy more silicon area than application specific analog integrated circuits. This is because of the larger number of transistors required for a digital circuit to perform the same operation as compared to an analog circuit. For example, the Xilinx Artix FPGA can have an area ranging from 100 mm^2 to 1225 mm^2 [115]. The NVIDIA GPUs have an area of 600 mm^2 [71] and the Movidius Myriad 2 has an area of 42.25 mm^2 [67]. These chip areas are very high compared to the proposed ASTCP chip which can be under 1 mm^2 (area increased to include pads, voltage supplies etc.) and hence the analog processor has a better performance per cost when compared to the digital hardware accelerators.

In Table 3 we also compare the chip area of various analog processors that have been reported in literature. Since the analog processors reported handle different frame sizes and have been fabricated in different technology nodes, we use a figure of merit (FOM2) to compare

performance. As can be seen from the table, the ASTCP has the lowest chip area and the highest FOM. Hence, it has the best performance per cost.

Table 3: Comparison of the proposed work with other designs reported in literature

Work Considered	Technology	Size of the Frame	Frames per Second	Power Consumption	FOM1[1]	Area of the Chip	FOM2[2]
[74]	0.6 µm CMOS	21x21	86K	37.5 mW	1.01e9	10 mm^2	1.01e8
[109]	0.35 µm CMOS	32x32	2000	600µW	3.41e9	2.7 mm^2	1.26e9
[108]	0.35 µm CMOS	1x1	100	0.066µW	1.51e9	1.2e-3 mm^2	1.25e12
[53]	0.25 µm CMOS	128x128	4000	20mW	3.27e9	16 mm^2	0.204e9
[48]	0.35 µm CMOS	160x120	2000	25mW	1.53e9	25 mm^2	0.061e9
[69]	90nm CMOS + Spin Based Devices	256x256	10000	220 µW	2.97e12	2.62 mm^2	1.13e12
Proposed Work	65nm CMOS	2560x1920	8000	7.92mW	4.96e12	0.28 mm^2	1.67e13

3.10.2. Reliability and Sensitivity of Analog Processor

The reliability of analog integrated circuits as compared to digital circuits is a concern. There are two major factors that affect the reliability of these devices. The first one is the sensitivity of analog circuits to variations in Process, Voltage and Temperature (PVT) and the second is the degradation due to aging mechanisms like bias temperature instability and hot carrier injection.

These effects have been studied extensively in literature and various methods have been devised to overcome them and perform a reliable analog and mixed integrated circuit design. In [117] the authors discuss using an efficient method for sizing of analog circuits for reliability and

[1] FOM1 = (Size of the Frame) x (Frames per Second) / Power Consumption
[2] FOM2 = FOM1/Area of the chip

they also study the trade-off between circuit lifetime and price for layout area. In [39] the author proposes an adaptive gate-source biasing scheme to improve the reliability of analog circuits. One of the techniques generally used to maintain an acceptable performance across all process corners and throughout the temperature range is to use compensation circuits. As an example, the performance of the integrator may drift due to temperature variation. The compensation circuit continuously measures the frequency response of the integrator and ensures that it always maintains a slope of 20dB/decade in the desired frequency range. If this varies due to temperature or voltage variations, the compensation circuit will adjust the bias of the integrator circuit and correct it. Methods like this can be investigated and used in designing the integrated circuit for the processor.

Noise in analog circuits can be very sensitive to temperature and process variations. Various techniques can be used to reduce the effect of noise and make the analog circuits less sensitive to noise. One method that can be used to reduce noise is using differential circuits. This provides a very good common-mode noise rejection and helps decrease the sensitivity of circuits to power and ground noise. Other techniques like limiting the circuit bandwidth, limiting external analog signals, providing signal separation and shielding can reduce the effect of noise in analog circuits. Another very effective technique used to reduce the noise figure of the signal chain is to have a low noise amplifier at the front end of the processor.

3.10.3. A Case Study: AlexNet Architecture

Convolutional Neural Networks (ConvNets) are a class of artificial neural networks that use massive network of neurons and synapses to extract local features from images. ConvNets are generally used for object detection and recognition applications. The main principle of a ConvNet is to extract features from feature maps of higher resolution and then combine these

features into feature maps of lower resolution. A general ConvNet follows two steps, a feature extraction step and classification step.

The feature extraction step consists of the following functions:

1. Convolution between an image of the previous layer and the corresponding set of weights, which produces an intermediate result for the feature map of the current layer. In this step features, such as edges, corners and crossings are extracted from input feature maps.
2. The next step is subsampling, which is done to achieve spatial invariance by reducing the resolution of feature maps. In this step a bias value is also added to the output maps.
3. In the next step, the output is passed through a non-linear activation function. This is applied to each pixel in the feature map.

To evaluate the performance of the analog space-time convolution processor we conduct a case study on AlexNet architecture [58]. It is a deep convolutional neural network model that was used to classify images into 1000 different classes. Figure 39 shows the architecture of AlexNet and Table 4 gives information about each of the five convolutional layers of the network. Layer 1 of the network has a 224x224 pixel image input with 3 channels. It is operated on by 3 kernels of size 11x11 which perform convolution. The outputs are subsampled and passed through a non-linear activation function. The resultant outputs are 55x55 pixel feature maps with 96 channels and it acts as the input to layer 2 of the network, where it is operated on by 96 kernels of size 5x5 which perform convolution. The outputs are again subsampled and passed through non-linear activation function. The output of layer 2 is the input to layer 3 and this process is continued in the other three layers of AlexNet as shown in Figure 39 and Table 4. The output of the fifth layer is sent to a classifier where the image is classified into one of the 1000 possible categories. These

applications are time consuming with high computational load. It is estimated that the 90% of the computation time in each layer of a convolutional neural network is spent on the first step which involves convolution of an image of the previous layer with a set of weights which produces an intermediate result for the current layer. Based on this assumption we calculated the total number of computations required in performing convolutions in the five layers of the AlexNet architecture and it was found to be 4.9 Giga operations. From this data, a performance comparison between the ASTCP and various other digital processors in the market was made.

Figure 40(a) shows the performance of various processors in GOPS/W [28]. The ASTCP processor is 8x faster than the closest convolutional neural network processor. Figure 40(b) shows the time required in milliseconds to process one 224 x 224 frame through the 5 layers of AlexNet architecture in each of the processors mentioned. An interesting point to note is that, for real-time operation the frame must be processed under 33.33ms. None of the processors apart from our analog processor (ASTCP) can achieve the real-time requirement.

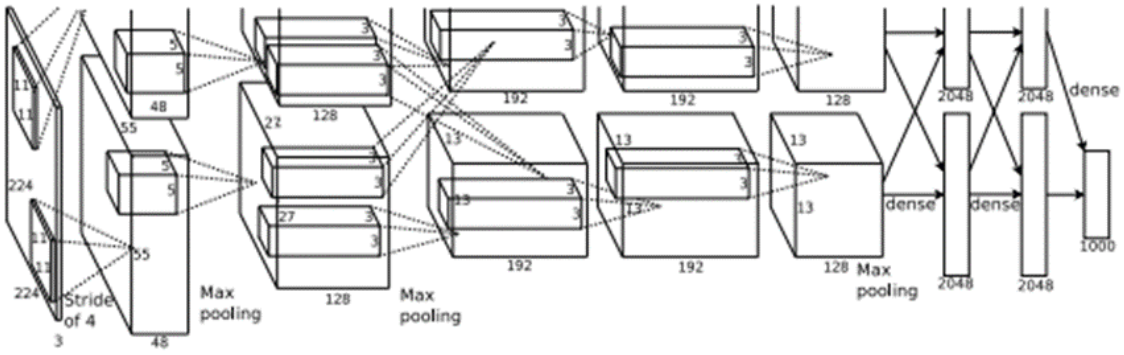
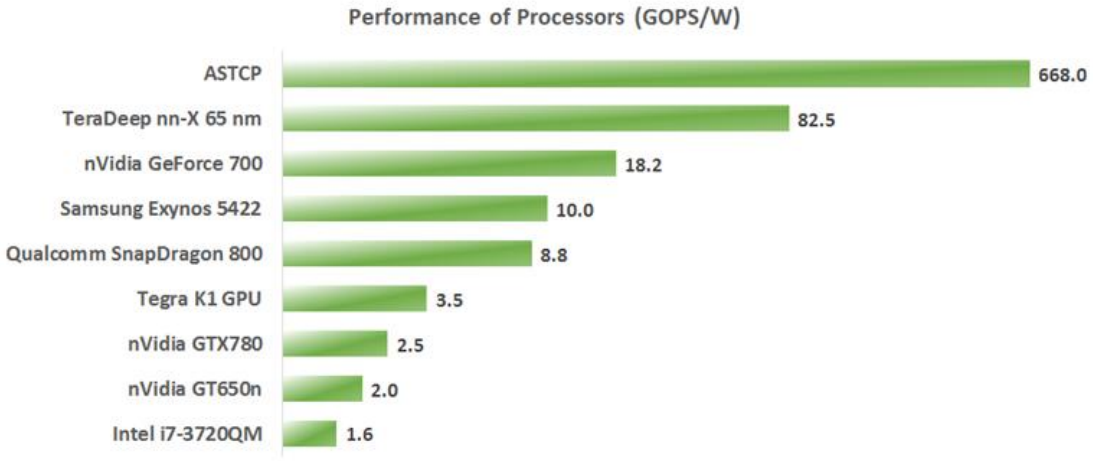


Figure 39: Architecture of the 5-layer AlexNet [58]

Table 4: Information about the five convolutional layers of AlexNet [58]

Layer	Number of Channels	Feature Map Size	Kernel Size	Number of Kernels
1	3	224x224	11x11	3
2	96	55x55	5x5	96
3	256	27x27	3x3	256
4	384	13x13	3x3	192
5	384	13x13	3x3	192

(a)



(b)

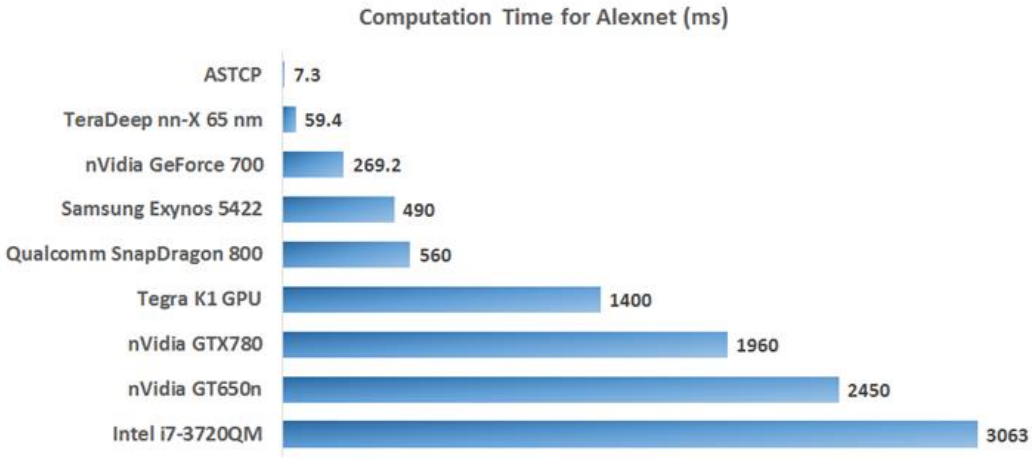


Figure 40: (a) Performance of various processors in GOPS/W (b) Time required in ms for processing one frame (224x224) in an Alexnet architecture

3.11. Conclusion

In this part of the dissertation, we developed a prototype of an analog processing board for machine vision and image processing applications. The analog processing board design was first simulated in ADS. Two frequently used image processing algorithms, RGB to YUV color transformation and edge detection, were used as benchmarks to compare the performance of the analog board against MATLAB implementation of these algorithms. Two performance metrics, PSNR and SSIM, were used for performance comparison and it was shown that the performance of the analog processing board is equivalent to a digital implementation. A PCB design of the analog processing board prototype was implemented, and it was shown that such a design has many advantages over a direct Integrated Circuit Design. The same benchmarking algorithms and performance metrics were used to measure the performance of the prototype board. The measured results indicate that the performance of the prototype is very close to simulated results in MATLAB. To perform speed and power consumption comparison, an integrated circuit design of the analog processor was proposed. Power consumption estimations were performed using previously designed and fabricated components or components taken from literature. It was shown that such an analog processor has a performance improvement of more than 100x over existing FPGAs and 5x improvement over state-of-art GPUs. The performance of the proposed integrated circuit design was also compared to other analog processors designed for machine vision applications reported in literature. A case study for a ConvNet implementation for object detection and recognition was performed and it was shown that ASTCP has 8x performance improvement over state-of-the-art digital processor. It was also shown that it is the only processor that can achieve real-time performance. As part of future work, we seek to investigate an analog processor composed of a combination of CMOS and emerging nanotechnology devices to further improve the performance.

CHAPTER 4

MEMRISTOR-CMOS ANALOG CO-PROCESSOR

In the previous chapter we demonstrated that a pure CMOS based analog processor can outperform many existing and projected digital and analog processors. This analog processor can perform very fast convolution and correlation type of operations. However, this processor was not designed to perform vector matrix multiplication. VMM computation underlies some of the major emerging applications. These applications require high computational speed and are run on platforms that are size, weight and power constrained. With the transistor scaling coming to an end, existing digital hardware architectures will not be able to meet this increasing demand. This opens an opportunity for custom hardware with novel device types and disruptive architectures to serve unmet needs. Research has shown that analog computation with its rich set of primitives and inherent parallel architecture can be faster, more efficient and compact for some of these applications. In this work, we propose the development of an analog co-processor to accelerate the solutions to HPC applications by invoking VMM in memristor-CMOS crossbar arrays. VMM computation underlies major HPC applications such as solving partial differential equations and performing deep learning inference and training. The practical realization of the long theorized 4^{th} fundamental circuit element [21][22], a memristor can change and "remember" its resistance state in an analog fashion as the current flowing through it is varied. This has important implications for an analog VMM. Once the matrix values are written into the memristor-CMOS crossbar array, the vector values are input on each row of the crossbar. The multiplication operation between the input voltage and the memristor resistance happens through Ohm's law and the accumulation of values along each column of the crossbar array is in the current domain, as explained by Kirchhoff's current law. The physical computation being

performed in the memristor-CMOS crossbar array provides a huge computational efficiency advantage over digital systems. Analog VMM has been a topic of research for many years and various circuit implementations have been developed ([59][66][14][32][33][45]), with machine learning and neural networks being the primary application areas ([13][91][92][19]). However, existing analog VMM based architectures can handle only fixed-point values and lower precision computation. In this work, we develop an analog co-processor architecture that can handle half precision floating point computation. The architecture is very modular and scalable. Most of the emerging applications have memory throughput issues mostly due to a large number of uncoalesced writes into memory. We propose a set of novel memory access techniques to mitigate this problem.

4.1. Introduction to Memristors

Memristive devices are electrical resistance switches that can retain a state of internal resistance based on history of applied voltage or current. These devices can store and process information and offer several key performance characteristics that exceed that of conventional integrated circuit technology. An important class of memristive device is a two-terminal resistance switch based on ionic motion, which is built from a simple conductor-insulator-conductor thin-film stack as shown in Figure 41.

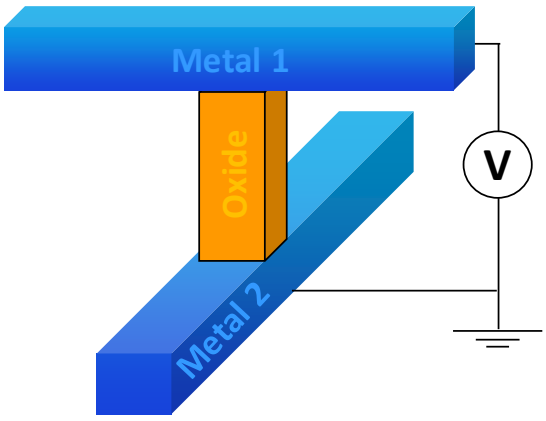


Figure 41: Structure of a typical memristor device consisting of an insulating layer sandwiched between bottom and top electrodes

4.2. SPICE Modeling of the Memristor and the Crossbar Array

An important requirement for using memristors in circuits is to have a model that can predict the behavior of the device which can be used in simulations and circuit design. The advantage of having a SPICE model is that it is compatible with most circuit simulation tools and this enables an easier design with memristors. The initial memristor model we have considered is the one developed at HP Research lab [96]. This model has been well calibrated with memristor devices built by the authors while at HP Labs. It was developed by performing a detailed study on electronic transport and switching dynamics data of the device. The typical memristor stack is Pt (15 nm) / Ta_2O_{5-x} (8 nm) / Ta (30 nm) ($0 < x < 1$). The Ta_2O_{5-x} is an amorphous film between two inert electrodes. This structure exhibits a bipolar resistive switching with a nearly linear low-resistance state and non-linear high-resistance state.

Let y denote the state variable for a TaO_x memristor device. The equations describing the static behavior of the device is given by Chua's memristor equation:

$$i(t) = G(y, v)v(t) \tag{26}$$

Where, $G(y, v)$ is the conductance for a given memristor state. In order to determine an approximate solution to $G(y, v)$, static measurements of current-voltage dependence were carried for different memristor states y.

The results reported from the HP paper [99] indicate that the key parameter to describe the memristor state is the oxygen content in the conduction channel. The state variable y is taken as a volume fraction of the channel with linear current-voltage relation and the remaining fractional (1- y) is insulating with a non-linear transport. The 'On' state transport was found to be

metallic and the 'OFF' state transport was best described by a non-linear Frenkel-Poole relationship. The device conductance is approximated by the parallel combination of the two phases as:

$$i = v\left[yG_m + (1-y)aexp\left[b\sqrt{|v|}\right]\right] \quad (27)$$

where G_m, a and b are constants for a particular device across all states and cycling history [96]. Based on this equation we create a static SPICE model for the device. In the next section, we present results of an ensemble of simulation data of the crossbar array for VMM operation.

4.2.1. Read Time of Memristors

One of the most important criteria for measuring the speed of the memristor is the read time of the device. The read time of each individual memristor directly translates to the speed of the VMM operation in a crossbar array. To demonstrate the read time of the memristor devices, we apply three voltage pulses of the same amplitude and pulse width of 0.5V and 100 ns respectively but with three different rise times of 0.1 ns, 1 ns and 10 ns. Figure 42 shows the output currents of the memristor device for these three input voltage pulses. As can be seen from the figure, the read time of the memristor depends directly on the rise time of the input voltage pulses being applied. This shows that the memristor can produce instantaneous outputs and the speed of operation is only limited by the design of the input driver circuits and the read-out circuitry at the output of the VMM crossbar array.

4.2.2. Resistance Ratio

There are various factors that affect the accuracy of the VMM outputs in a memristor-CMOS crossbar array. One major factor is the ratio of the memristor resistance to the resistance

of the wire connecting the memristors. It was observed that higher ratios provided better accuracies.

Figure **43** shows a plot of the percentage error of VMM output versus the resistance

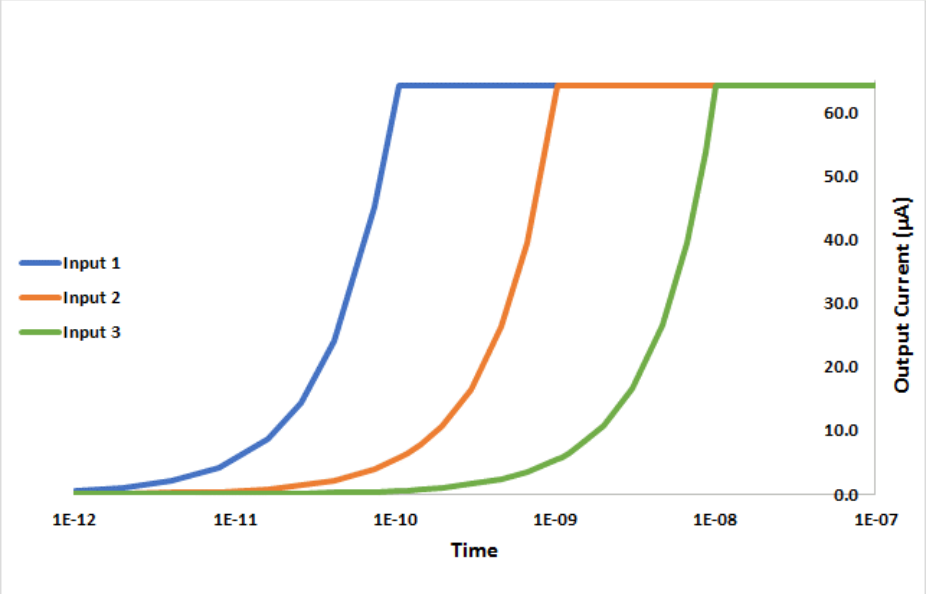


Figure 42: Dependence of memristor read time on the rise time of the driver

ratio in a 4 x 4 array. As can be seen from the figure, error percentage is less than 1% for a ratio of 10^3 but as the ratio decreases the error increases. The resistance ratio also depends on the size of the array. As the size increases, the resistance ratio that must be maintained to get the same value of error at the output increases. For a 16 x 16 and 32 x 32 array, resistance ratios of 2500 and 6300 are required to maintain the error below 1%. Another factor that can affect the accuracy of VMM outputs is the non-linearity of the memristor. Memristors tend to be non-linear towards their high resistance state. Hence, for VMM operation it is critical that memristors are designed such that all resistance states are in the linear region which are towards the low resistance state.

4.2.3. Sneak Path Currents

Sneak paths are undesired paths for current, parallel to the intended path. The full write voltage V_{wr} for the memristor device is applied to the row of the crossbar or is achieved by applying $0.5V_{wr}$ on the row and $-0.5V_{wr}$ on the column of the crossbar array. The memristor at the junction of this row and column becomes a fully selected device. However, other memristor devices sharing the same row or column which are not being written into have a smaller voltage

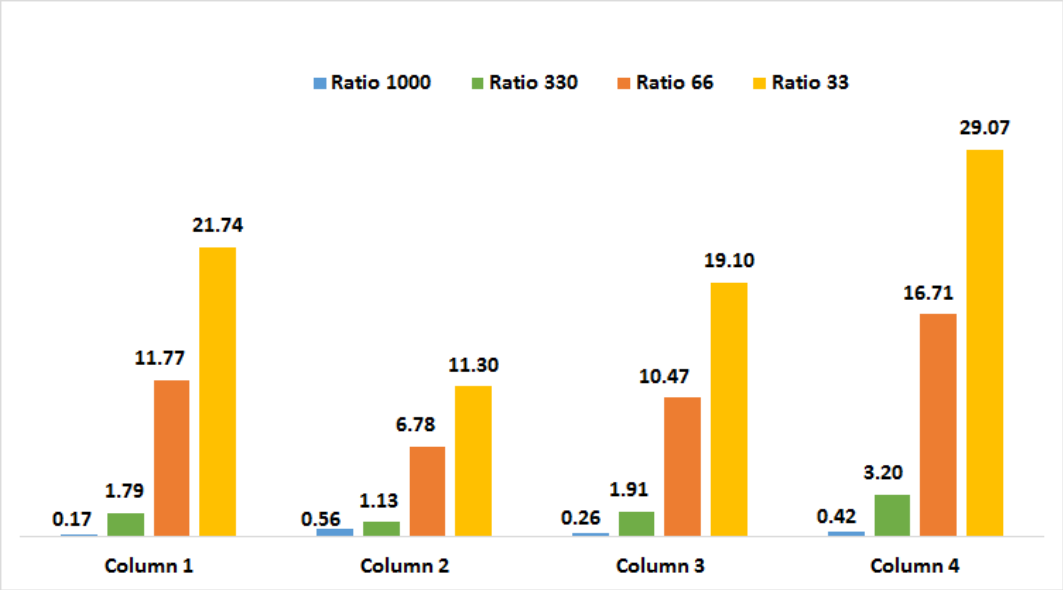


Figure 43: Variation of percentage error of output current along each column with increasing wire resistance for a 4x4 memristor crossbar array

drop, for instance $0.5V_{wr}$ in the case. These memristors are called half selected devices and a current (of the order of mA in this example) will flow through them. These currents are called sneak path currents. A transistor connected to the row and column of the crossbar array can be used as a selection device to mitigate the sneak path currents. A crossbar array with a transistor device in series with a memristor at each cross point is called a 1T1R architecture. The transistors that are connected to the memristor devices that are not being written into will remain in the OFF state, which will greatly reduce the sneak path currents. A 0.25µm TSMC model of a transistor

was used in this work and the sneak path currents were measured in an 8 x 8 crossbar array with the memristors in the first column of the array selected and the result is shown in Figure 44. The selected memristors have high current flowing through them as expected. The non-selected devices in the other columns have a current less than 1.5pA flowing through each device which is a huge reduction.

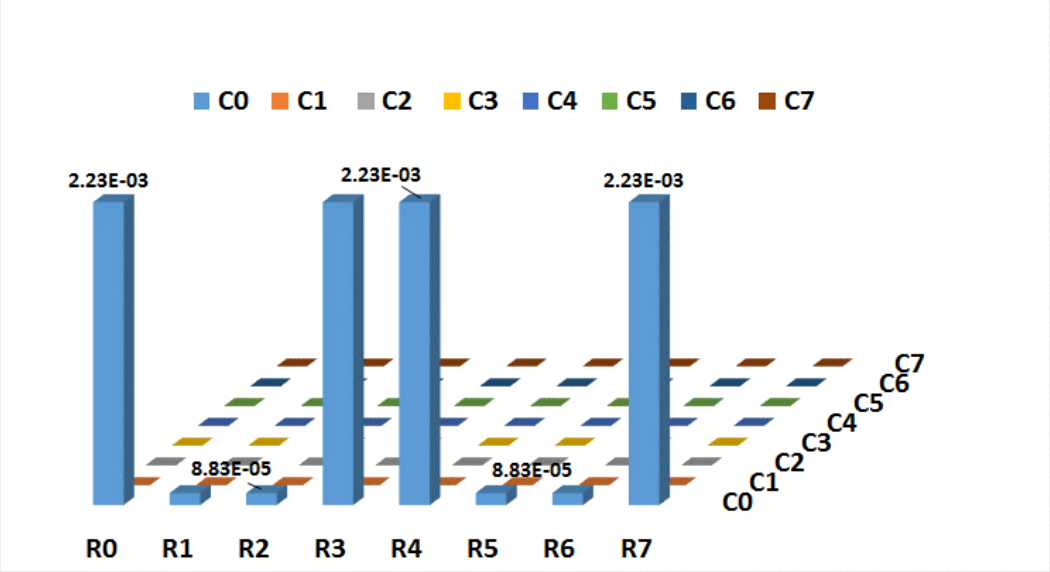


Figure 44: Output currents measured at each memristor for an 8 x 8 1T1R array with all the memristors in the first column selected

4.3. Memristor-CMOS Crossbar Array Architecture

Figure 45 shows the architecture of a memristor-CMOS crossbar array used in the analog co-processor. The architecture consists of a M x Marray of memristors and transistors arranged in a crossbar configuration. The crossbar array is used to perform multiplication of a vector $\vec{x}$ with a matrix *A* to produce an output vector $\vec{y} = A\vec{x}$.

4.3.1. Memristor Write Operation

The first step of performing a VMM operation is to store the matrix values in the memristor crossbar array as conductance of the memristors. The memristors used here have a

High Resistance State (HRS) and a Low Resistance State (LRS). The highest and lowest values of the matrix *A* are first identified and are then mapped to LRS and HRS of the memristor cell respectively. All the other matrix values are linearly mapped to be between the HRS and LRS of

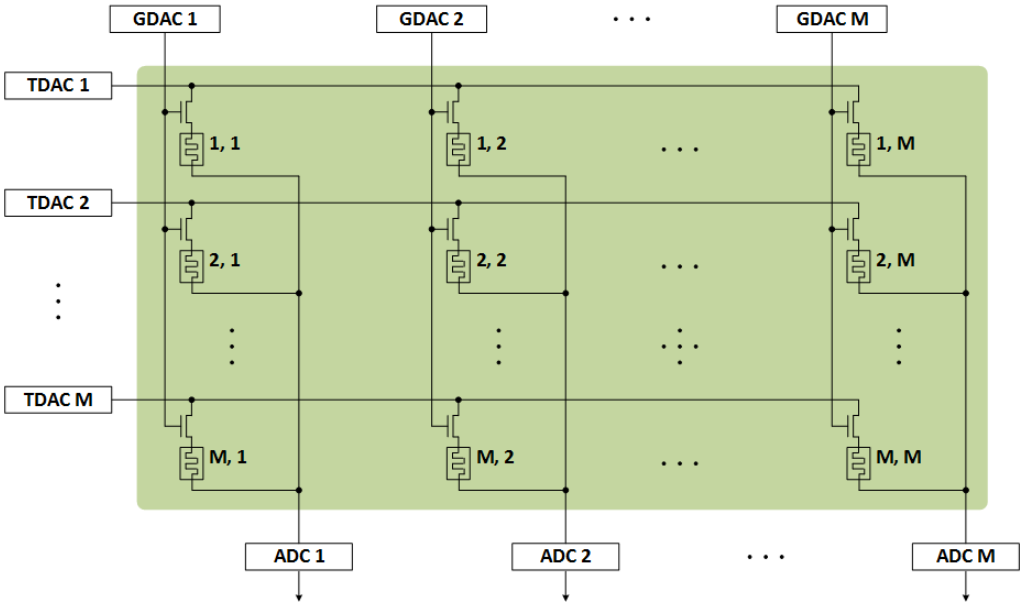


Figure 45: Architecture of the memristor-CMOS crossbar array

the memristor. Once the matrix values have been mapped to memristor resistance/conductance values, these must be stored in the memristors of the crossbar array. This is accomplished by performing memristor writes.

Here we have assumed that each value of the matrix *A* can be stored on a single memristor of the crossbar array. A memristor can be written into by applying a voltage pulse of a given amplitude and pulse width, which sets the memristor to a conductance state. A write-verify-write programming cycle can be used to write conductance values into the memristor to within 1% tolerance of the target conductance values as described in [45]. To enable this operation, each row of the memristor crossbar array has a Digital-to-Analog Converter (DAC) as shown in Figure 45. The GDACs are connected to the gate terminal of the transistors. These DACs can be used to input a gate voltage pulse which can turn the transistors 'ON' and 'OFF'. They can also be used to

set the compliance current of the memristors, which is defined as the maximum current that flows through the memristor. Setting the compliance current helps to achieve better control over memristor during the write operation. The TDACs are used to apply a write voltage pulse to the top electrode of the memristor. This sets the memristor to a conductance state, accomplishing the write operation.

4.3.2. Memristor Read Operation

Once the matrix values of *A* have been written into the memristor-CMOS crossbar array, it can now be used to perform VMM operation. The VMM operation is performed through memristor reads. Here the values of the input vector $\vec{x}$ are converted into voltages. These voltage values are input along each row using the TDACs. The input voltage values multiplied with the memristor conductance results in an output current at each memristor junction which are then summed along each column. These currents which represent the VMM outputs are converted to voltages and then digitized using Analog-to-Digital converters (ADCs).

4.4. Analog Co-processor Architecture

The analog co-processor that is proposed in this work will be used for applications requiring real time computation on size, weight and power constrained platforms. The co-processor will be used to accelerate applications that require high computational efficiencies by using VMM as the core computation. VMM computation is used in a variety of applications ranging from simple image filtering operations to more complex problems in machine vision, deep learning and scientific simulation. Since the analog co-processor will be used in a variety of application environments, it is important that we interface it to a standard bus interface. The analog co-processor will work with a digital processor and it will also have other support circuitry

to perform non-VMM computations and for data movement.

4.4.1. The VMM Signal Processing Chain

The analog co-processor is designed to work in conjunction with a digital processor. The digital system running an application or algorithm will identify a computationally intensive task which will be offloaded to the analog co-processor. The analog co-processor takes over and performs VMM computation in the memristor-CMOS crossbar array. However, there are various steps leading to the VMM computation and there are steps after the VMM computation in the signal processing chain that are as critical as the VMM computation itself.

4.4.1.1. Data Formatting for the Analog Co-Processor

The analog co-processor can be used for a variety of applications. These applications perform different kinds of matrix multiplication operations which must be cast into a VMM dot product framework to be implemented on the analog co-processor. Depending on how the data is handled by the analog co-processor, the matrix operations can be divided into two major categories. The first class of matrix multiplication operations are the general matrix-matrix multiplication operations. In these operations, one of the matrices is written into the memristor crossbar array as its conductance values and the other matrix is read out row wise and transformed into a column vector and multiplied using the memristor-CMOS crossbar array. The second class of matrix multiplication operations are 2D convolutions and correlations. The 2D convolution and correlation operations can be cast into a VMM framework but these require a shift and add operation to be performed along with a matrix multiply. Here a matrix of values is written into the memristor crossbar array as their conductance values. The other matrix is read out in an overlapping sequence to represent the shift required for a convolution operation. The

outputs of the memristor crossbar array along each column have to be further summed to obtain the final 2D convolved result. Which of the two major classes of matrix multiplication operation is to be processed by the analog accelerator in a given instance will be distinguished by the application layer. The application will include a header field which will mark the matrix data to be sent to digital memory as pertaining to one or the other category. This data representation will then be sent into digital memory, directly preceding the corresponding matrix data. The data representation can be regarded as a pseudo instruction and can be formatted as shown in Figure 46. Here, the opcode identifies the class of matrix multiplication, matrices A and B are the operands undergoing VMM and, the return target is used to store the result. All matrix values are stored contiguously by row and can be accessed by referencing their start and end address.

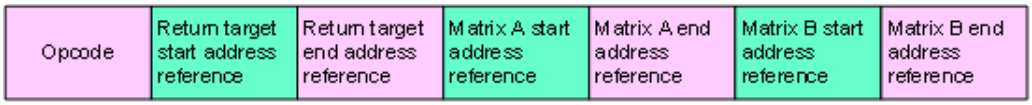


Figure 46: Format for pseudo instruction

4.4.1.2. Normalizing and De-Normalizing Blocks for Handling Floating Points Numbers

Different applications being implemented on the analog co-processor require different levels of precision and dynamic range. In order to accommodate this, a floating point format is used. Floating point is a representation of real numbers that trade-off between range and precision. In a floating point representation, the number is represented by a fixed number of significant digits called the significand and scaled using an exponent in some fixed base. The general representation is of the form: significand x $base^{exponent}$. A variety of floating point representations have been used over the years, but since the 1990s the most commonly used floating point representation is as defined by IEEE754 standard.

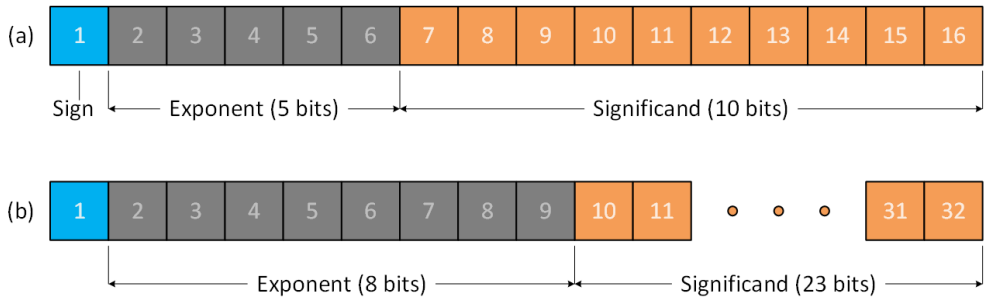


Figure 47: (a) IEEE754 half precision (16 bit) floating point representation, and (b) IEEE754 single precision (32 bit) floating point representation

Figure 47(a) shows the IEEE754 representation of a half precision floating point representation of a real number and Figure 47(b) shows the IEEE754 representation of a single precision floating point representation of a real number. The sign bit determines the sign of the number being represented. A '0' on the sign bit represents a positive number and a '1' represents a negative number. The exponent for the 16-bit floating point is a 5-bit unsigned integer from 0 to 31. The actual exponent value used is the exponent value with a bias subtracted. The bias value for a 16-bit floating point representation is 15. So, the effective value of the exponent ranges from -15 to 16. The true significand includes 10 fraction bits to the right of the binary point with an implicit leading bit. Only 10 fraction bits are stored in memory, but the total precision is 11 bits. This corresponds to 3.311 decimal digits.

The exponent for the 32-bit floating point is an 8-bit unsigned integer from 0 to 255. The actual exponent value used is the exponent value with a bias subtracted. The bias value for a 32-bit floating point representation is 127. The effective value of the exponent ranges from -127 to 128. The true significand includes 23 fraction bits to the right of the binary point with an implicit leading bit. Only 23 fraction bits are stored in memory, but the total precision is 24 bits. This corresponds to 7.22 decimal digits.

The binary representation for the 16-bit floating point number is given by the following equation:

$$(-1)^{b_{15}} \text{ x } (1.b_9b_8 \dots b_0) \text{ x } 2^{b_{14}b_{13}\dots b_{10}-15} \tag{28}$$

This yields a decimal value of:

$$(-1)^{sign} \text{ x } \left(1 + \sum_{i=1}^{10} b_{23-i}2^{-i}\right) \text{ x } 2^{e-15} \tag{29}$$

The minimum normalized positive value that can be represented using a 16-bit floating point representation is $2^{-14} = 6.1\text{x}10^{-5}$ and the maximum value is $(2\text{-}2^{-10}) \text{ x } 2^{15} = 65504$.

The minimum normalized positive value that can be represented using a 32-bit floating point representation is $2^{-126} = 1.18\text{x}10^{-38}$.

Consider two floating point numbers *X* and *Y*. The significands of *X* and *Y* are denoted as X_s and Y_s and the exponential parts are denoted as X_e and Y_e respectively. The floating-point number can be added as follows:

1. Convert both numbers into scientific notation by explicitly representing the '1'.
2. In order to add these numbers, the exponents should be the same. This step is achieved by shifting the radix point of the mantissa. This process is called normalization.
3. Add the two mantissas.
4. Adjust the result and represent it as a floating-point number.

The multiplication of *X* and *Y* is then given by:

$$X.Y = (X_s.Y_s)2^{X_e+Y_e-15} \tag{30}$$

An innovative procedure to perform floating point VMM computation has been developed here. Floating point values differ from fixed point values in that they are represented by a number (mantissa) raised to a power (exponent), with a preceding sign bit. Floating point

numbers are handled by normalizing the exponents and aligning the mantissas. This is more efficient when performed on a set of values. First step of normalizing the exponents generally involves identifying the exponent 'E_{max}' of the extreme value. Other values are then normalized to have the same exponent E_{max} by bit-shifting them by a factor (E_{max}-E-Bias), where E is the exponent of each individual value and the 'Bias' term is a constant for a particular bit precision (e.g. Bias is 127 for a 32-bit floating point number). In the worst case, the normalizing factor can be as large 278 bits for single precision floating point computation. To circumvent this problem, elements of the VMM array are aligned by taking advantage of the fact that the difference between the exponents of the neighboring elements is significantly less than the extreme values. The value of E_{max} is stored, to be used during the de-normalization process.

4.4.1.3. Mapping and Inverse Mapping Blocks of the Analog Co-Processor

The first step of performing a VMM operation is to store the matrix values in the memristor crossbar array as conductance of the memristors. We use a mapping procedure similar to the one developed by in [45]. The memristors have high conductance state (Y_{max}) and a low conductance state (Y_{min}). Let the matrix to be mapped into the memristor crossbar array be *A*. The highest and the lowest values of the matrix are first identified as A_{max} and A_{min}. Two linear mapping coefficients are then defined as $a = \frac{Y_{max} - Y_{min}}{A_{max} - A_{min}}$ and $b = Y_{max} - a(A_{max})$. Using the two coefficients, the elements of the matrix *A* are mapped to the memristor conductance values as $Y = a.A + b$. This mapping works for negative numbers as well. After the elements of the matrix have been converted into memristor conductance values, these are further converted into write voltages for the memristors. During the write operation, these write voltages are used to store the conductance values in the memristors of the crossbar array using the TDACs shown in Figure 45. The input vector values are also linearly mapped to memristor read voltages and are input

into the crossbar array using TDACs shown in Figure 45, during the read operation. After the completion of the VMM operation, the output voltages must be inverse mapped to actual values. If the input vector is *x* and the output vector is *V,* then the inverse mapping operation is accomplished as $O = \frac{(V - b*sum(x))}{a}$.

4.4.2. Multi-Bit Precision Capability

Different applications require different bit precisions. For example, image processing applications may only need 8 bits of precision. Applications in machine vision and deep learning may need up to 16 bits of precision whereas most scientific simulation applications demand single precision computation. Hence, it is critical that the analog co-processor that we are designing has multi-bit precision capability and this is achieved through a bit sliced approach of handling data.

In section 4.3, we had assumed that each memristor can store the entire value of one matrix element. However, in practice each memristor can store a limited amount of information. It has been demonstrated that the TaO_x based memristors can store up to 5 bits of information [45]. In order to write a value into the memristor cell very precisely, a write-verify-write programming cycle is used. A higher precision requirement generally means more number of write cycles.

Consider a matrix *A* of size *N* x *N* which is to be written into the memristor crossbar array of size *M* x *M* as shown in Figure 48. Assume that this matrix is made up of *n* bit numbers. The matrix is first divided into smaller matrices whose size is either equal to or less than the size of the memristor crossbar arrays ($\frac{N}{M} \text{x} \frac{N}{M}$ in this case). Let us further assume that each memristor can store *m* bits of information, where *m* < *n*. To write each *n* bit element of one $\frac{N}{M} \text{x} \frac{N}{M}$ dimensional submatrix into the memristor-CMOS crossbar array, the elements have to be bit sliced. Hence,

the total number of crossbar arrays required to write one $\frac{N}{M}$x$\frac{N}{M}$ dimensional submatrix is *n/m*.

The input vectors that are used to perform VMM operation are supplied by the TDACs shown in Figure 48. If the resolution of the TDACs are *m* bits and the input vectors are all *n* bits then a bit sliced approach is further required to perform VMM, where each TDAC supplies *m* bit vector input. This would mean that, in order to perform VMM on a $\frac{N}{M}$x$\frac{N}{M}$ dimensional submatrix we would need $(n/m)^2$ crossbar arrays.

To understand this better let us consider an example. Consider a matrix *A* of size 32 x 32. Let the size of the memristor-CMOS crossbar array be 16 x 16. Let us further assume that the matrix *A* is made up of elements each of which are 8 bits and that each memristor can store up to 4 bits of information. The matrix *A* is first divided into 4 matrices of size 16 x 16, with each submatrix containing elements which are 8 bits wide. To write the values of one such submatrix we need two memristor-CMOS crossbar arrays as the memristors can store only 4 bits of information. So, for 4 submatrices we need a total of 8 memristor-CMOS crossbar arrays.

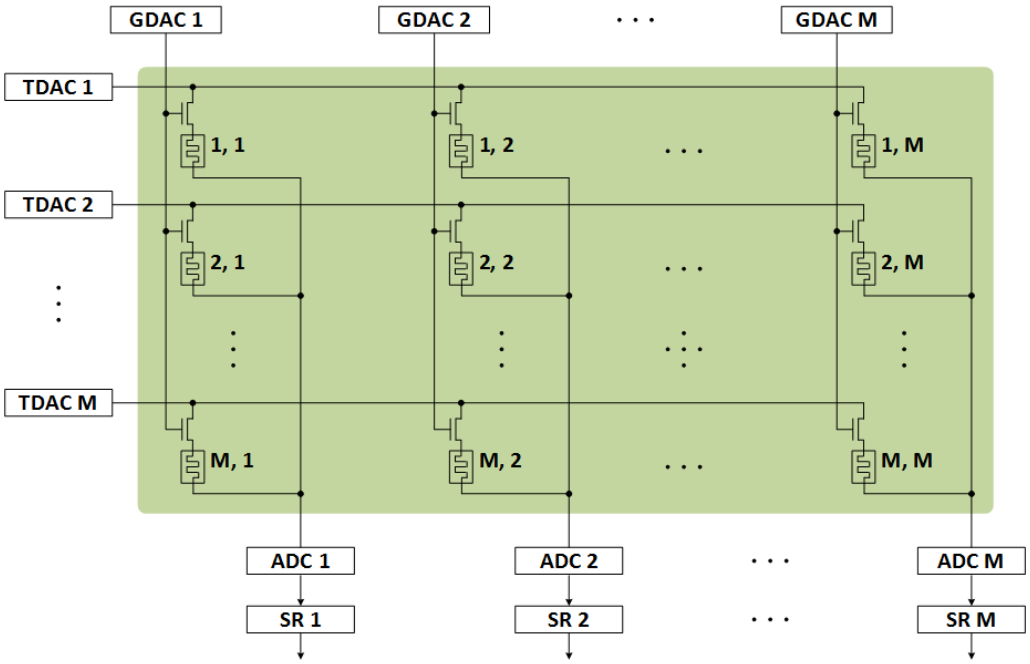


Figure 48: Architecture of the analog co-processor cell

Let us now consider the VMM operation. If the resolution of the TDACs is assumed to be 4 bits and the elements of the input vector are assumed to be 8 bits then we need a bit sliced approach for performing VMM. Hence the input vector is split into two input vectors, each of which has elements that are 4 bits wide. This would mean that we need one more set of 8 memristor-CMOS crossbar arrays (16 total) which have the same information stored in them as before, to complete the VMM operation.

Since we use a bit sliced approach to achieve higher bit precision, the results of performing VMM on each submatrix and bit sliced vector must be shifted, summed and then accumulated to get the correct result. Figure 48 shows the complete architecture of a single VMM cell of the analog co-processor. The VMM cell shown here can be considered as the computational building block of the analog co-processor. In addition to the memristor-CMOS crossbar array and the converters as shown in Figure 45, the VMM cell has shift registers at the output of the ADCs to shift the outputs to their correct bit position. The shift registers of each VMM cell shift the outputs by a fixed, pre-assigned value.

4.4.3. Design of the Core and Engine of the Analog Co-Processor

The analog co-processor being designed can handle problems of different dimensions, sizes and precision requirements. As mentioned in the previous section, multiple VMM cells are used to achieve a higher precision computation. We refer to a group of VMM cells that store the bit slices of the same submatrix along with other support circuitry as a core. Figure 49 shows an example architecture of a single core of the analog co-processor with four cells. Multiple cores are required to perform a complete VMM operation as they operate on multiple submatrices of a single input matrix. We refer to this collection of cores as an engine. Figure 50 shows an example

architecture of an engine of the analog co-processor.

The input matrix and vectors are divided into submatrices and sub vectors and are stored in the external memory. The engine of the analog co-processor is connected to external memory though a Direct Memory Access (DMA) Engine which will be fed by a high speed Serializer/Deserializer (SerDes) link [60]. Depending on the input bandwidth required, each core may be supplied by its own SerDes link. In this case, each core would have its own DMA engine and each cell would have its own input scratchpad memory. The floating point values of each input submatrix and sub vector will be normalized and then mapped to the appropriate conductance and voltage values as explained in the previous section by the normalizing block and the mapping block of the engine shown in Figure 50. These mapped conductance values and input vector are distributed to each core as shown by the green input bus of Figure 50. All the VMM

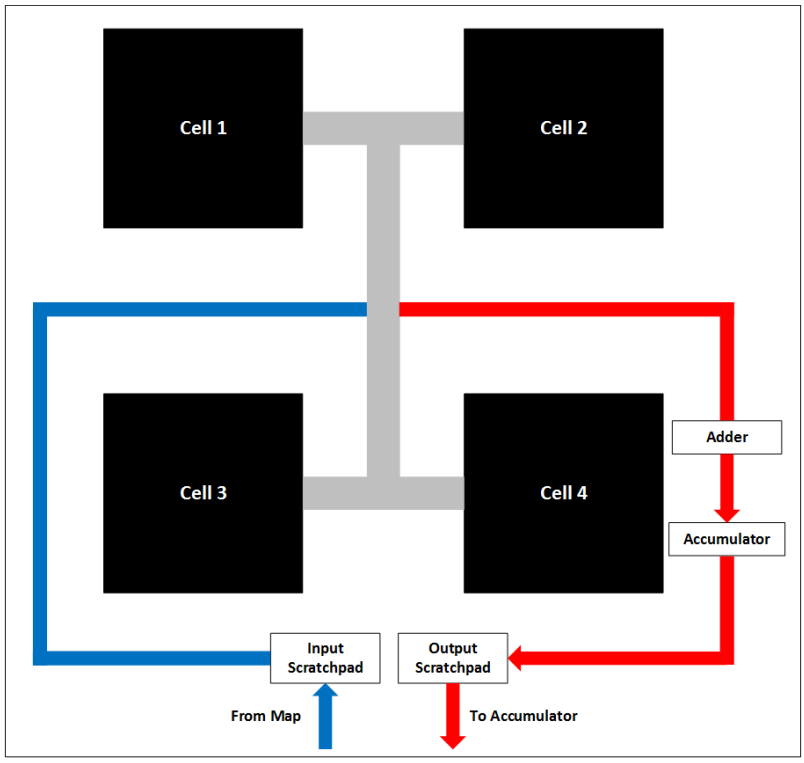


Figure 49: Architecture of the analog co-processor core

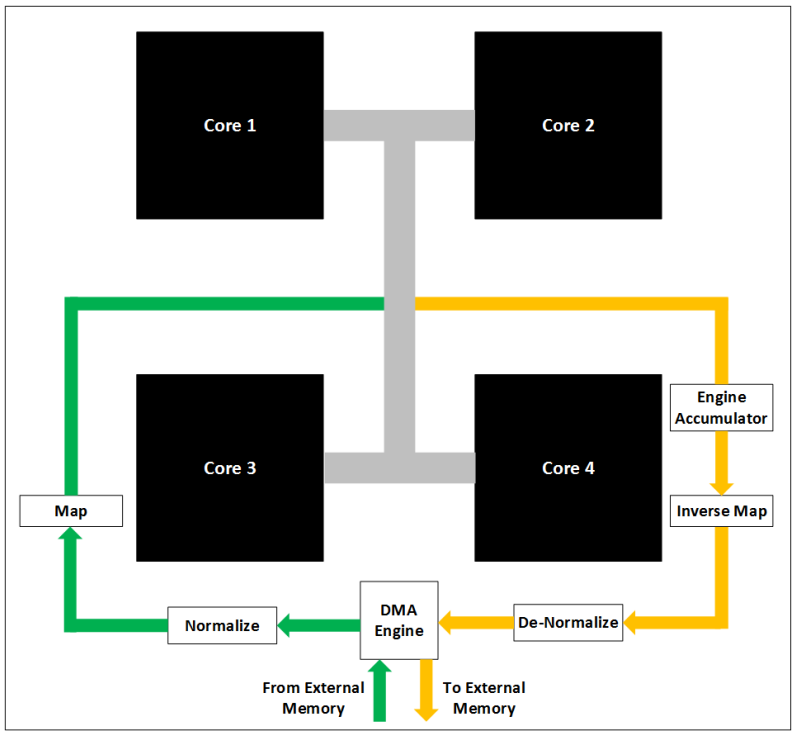


Figure 50: Architecture of the analog co-processor engine

cells in a core are operated on by the same input vector and hence these cells can share the same read DACs (TDACs). The number of cells per core in our architecture is determined by the ratio $(n/m)^2$ where n is the number of bits of the elements of the matrix and the input vector, and m is the number of bits per memristor cell. The other factors that affect the maximum number of cells per core are the voltage drops due to the DACs being shared and the input bandwidths to perform VMM operation. The mapped conductance values of the submatrix and the input vector values are stored in the input scratchpad as shown in Figure 49. The scratchpads will be based on a SRAM design. The key benefits of using SRAM over DRAM are the enormous reduction in memory access latencies and power consumption. Since a bit sliced approach is being used in the analog co-processor, one important criteria to ensure the correct functionality of the co-processor is to operate all the cells in lockstep i.e. all the VMM cells of the co-processor should receive inputs at the same time and the outputs of the cells should be sampled, processed and stored at the same

time. To ensure this, all the cells of the core are connected through a H-tree network. H-tree is a commonly used interconnection network that routes signals to different parts of a chip with equal propagation delay. As can be seen from the figures, the cores in an engine and the cells in a core are interconnected through a H-tree network. The outputs of each cell are from the shift registers as shown in Figure 48. Since these are bit sliced outputs, they have to be added and then accumulated to obtain the final VMM result. The outputs of all the cells are sent to an adder block as shown in Figure 49. The adder block is made up of an adder tree network for multiple stages of addition. The output of the adder is sent to an accumulator block which combines the VMM outputs from multiple cells and stores the intermediate results in the output scratchpad memory. The outputs from the scratchpad memory of multiple cores are sent to the engine accumulator as shown in Figure 50. The engine accumulator accumulates outputs from multiple cores and the output of the accumulator goes to the inverse mapping block. The inverse mapping block converts the output voltages back to actual values. These values are converted back to floating point representations by the de-normalizing block and are sent to external memory for storage through the DMA engine and the high speed SerDes link.

4.4.4. Memory Management for the Analog Co-Processor

One of the most significant challenges in the design of the analog co-processor is ensuring sufficient memory bandwidth to the device to fully realize the benefits of VMM through memristor crossbar arrays. To ensure this, the following memory management techniques are used.

4.4.4.1. Inherent Threading

The analog co-processor allows for 'inherent threading' by enabling concurrent processing per crossbar row/column. Each row of the input matrix can be assigned to a set of crossbar columns for VMM operation, similar to the way a CPU assigns a separate thread to each row. The matrix row can be operated in parallel by all the crossbars in an engine, similar to a SIMD execution. This simplifies thread management and streamlines VMM processing relative to that performed in digital systems.

4.4.4.2. Scratchpad Memory

Individual blocks of on-chip SRAM are assigned to crossbar cores within the analog co-processor cores as shown in Figure 49. The purpose of these SRAM banks is to serve as a fast scratchpad memory for VMM computation. The key benefit of SRAM over DRAM for storing matrix values and partial results is the enormous reduction in memory access latency it offers, on the order of 10-100 times. There is also a complementary reduction in power consumption, from 640 pJ per off-chip LPDDR2 DRAM access to 5 pJ per on-chip SRAM access [40].

4.4.4.3. Novel Data Representation to Facilitate VMM Updates

The analog co-processor's primary function is to accelerate VMM, and high throughput is required to keep its crossbar cores busy. The use of a novel data representation can facilitate updates to keep throughput high, while minimizing expensive write operations to crossbar cores. To accomplish this objective, column vectors are encoded in a format that includes a flag bit and unique identifier as shown in Figure 51. The unique identifier can be cell number, which is followed by core number and engine number. In this new data structure, if the column vector

has changed and needs to be updated, its flag bit will be set, and an interrupt will be triggered to write the new value into the crossbar cores of the corresponding floating point engine.

Figure 51: Data representation associated with each matrix column vector

4.4.5. Computational Efficiency of the Analog Co-Processor

The computational efficiency of the analog co-processor is calculated theoretically by considering the power consumption of various components present in the analog co-processor. Table 5 gives the power consumption and area of the components in the analog co-processor. These numbers are based on previously designed and fabricated components as indicated by the references shown in the table. The digital circuit incorporates normalize, de-normalize, map, inverse map and accumulator blocks. The power consumption and area of the analog co-processor has been computed for single precision floating point computation. The size of each memristor-CMOS crossbar array is assumed to be 128 x 64. The total number of cells required to perform a 32-bit computation is 25, with each memristor storing 5 bits of information. This would mean that the total number of engines in the analog co-processor would be 2, with each engine having 4 cores and each core having 4 cells. Each 128 x 64 memristor crossbar array can perform 8192 multiplications and 8128 additions, which corresponds to 16320 total operations while consuming a power of 3.04W. The frequency of operation of the analog co-processor is assumed to be 250MHz for these computations. This gives a computational efficiency of 1.34 Tera operations per second per watt (TOPS/W) for a 32-bit floating point operation. The computational efficiency for 16-bit and 8-bit precision are 8.4 TOPS/W and 31.18 TOPS/W respectively. The total area of the analog co-processor is estimated to be 228.71 mm^2, 36.68 mm^2 and 9.59 mm^2 for 32-bit, 16-bit and 8-bit precisions respectively.

Table 5: Power consumption and area of the components of the analog co-processor

Components	Power Consumption for Write (mW)	Power Consumption for Read (mW)	Area (mm^2)
DACs [27]	312.5	234.4	102.4
ADCs [18]	1102.3	1168.9	90
Crossbar Array	19.6	1117.1	0.01
Digital Circuit [2][45][43][1][33]	454.65	509.71	36.14
Scratchpad [20]	1.2	13.7	0.16
Total	1890.25	3043.8	228.71

4.4.6. Performance Comparison of the Analog Co-Processor

Based on power consumption computed in the previous section, we calculate the performance of the analog co-processor and compare it to the projected performances of state-of-the-art digital processors assumed to be developed in 5nm CMOS technology in the year 2022 and, other analog and digital processors that have been developed or proposed in literature. The metric used for performance comparison is TOPS/W.

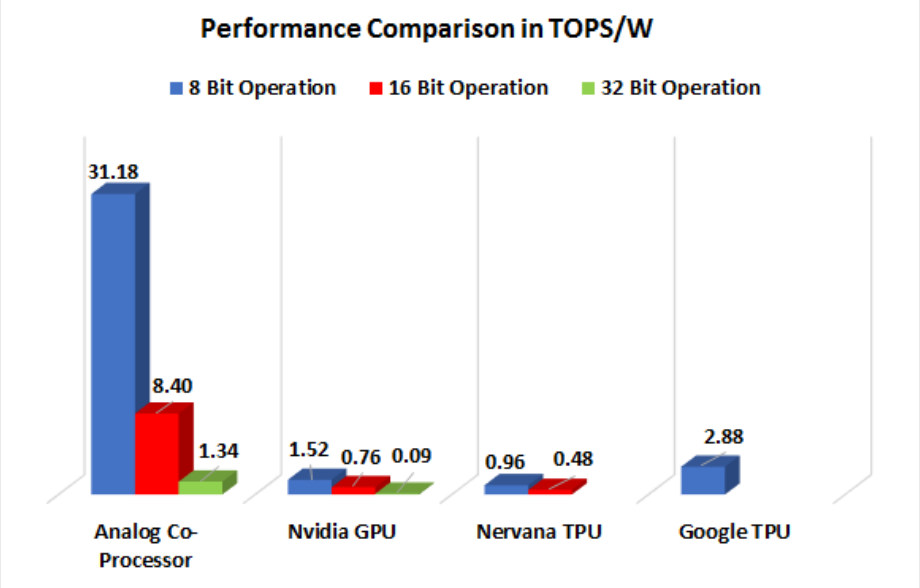


Figure 52: Performance comparison of the analog co-processor with existing state-of-the-art digital processors in TOPS/W

Table 6: Comparison of the proposed work with other processors reported in literature

Processor	Technology	Power (W)	Area (mm^2)	Computational Efficiency (TOPS/W)
DaDianNao	28nm CMOS	20.1	88	0.286 (fixed)
Rex Neo	16nm CMOS	8	100	0.256 (float)
ASTCP (core)	65nm CMOS	7.9e-3	0.28	0.334 (fixed)
ISSAC	Memristor	65.8	85.4	0.644 (fixed)
Proposed Work	Memristor	3.043	36.68	8.4 (float)

Figure 52 shows the performance comparison of the analog co-processor to three existing digital processors namely, Google Tensor Processing Unit (TPU) [52], Nervana TPU [46] and Nvidia GPU [72]. The performance is compared for three modes of computation, 8-bit, 16-bit and 32-bit precision. The Google TPU can only perform 8-bit computations. The Nervana TPU can only perform 8-bit and 16-bit computations. Nvidia GPU can support all three bit precisions. As can be seen from the figure, for an 8-bit precision the performance of the analog co-processor is 11x better than the Google TPU, 32x better than the Nervana TPU and 21x better than a Nvidia GPU. For a 16-bit floating point computation the analog co-processor is 18x better than the Nervana TPU and 11x better than the GPU. For a 32-bit floating point computation the analog co-processor is 15x better than the GPU.

Table 6 compares the performance of the analog co-processor to various other CMOS and memristor based processors reported in literature. The first processor we consider is DaDianNao [100]. It is a 28nm digital processor built to accelerate Convolutional Neural Network and Deep Neural Network algorithms for machine learning. It is designed to operate in 16-bit fixed point mode and can achieve a peak computational efficiency of 0.286 TOPS/W. Rex Neo [94] is a 16nm

digital processor built for supercomputers which can achieve a peak performance of 0.256 TOPS/W of 16-bit floating point computation. In a prior work, a CMOS analog processor for machine vision applications called ASTCP was designed and a prototype of the processor was implemented. It was shown that such a processor would achieve a performance of 0. 668TOPS/W for 8-bit fixed point computation [12]. ISSAC [90] is a memristor based analog Convolutional Neural Network accelerator that achieves 0.644 TOPS/W for 16-bit fixed point computations. In comparison to these processors the analog co-processor that is proposed in this work has the best performance and can achieve a peak performance of 8.4TOPS/W of 16-bit floating point computation. There is no published literature, to our knowledge, of any other memristor based analog co-processor that can perform floating point computation.

A good processor design should always achieve an optimum cost-performance tradeoff at a particular target performance. There are two parts to the cost of a chip. The first one is a fixed cost also known as the Non-Recurring Engineering (NRE) cost which accounts for the chip design, verification and mask generation. The second one is the recurring cost which is proportional to product volume and it includes the chip area, packaging and test. The NRE cost always keeps increasing and multiple design runs of a chip adds to this cost. This will be addressed by first designing a prototype as was demonstrated in our prior work [12]. Once the design has been finalized we can move into an integrated chip design phase, which will now be cheaper. The recurring cost can be reduced by having a smaller chip area. The digital hardware accelerators generally occupy more silicon area than application specific analog integrated circuits. This is because of the larger number of transistors required for a digital circuit to perform the same operation as compared to an analog circuit. For example, the NVIDIA GPUs have an area of 600 mm^2 [71] and the Google TPUs have a core area $\leq$ 331 mm^2 [52].These chip areas are higher compared to the area of the proposed analog co-processor which is around 36.68 mm^2 and hence

the analog processor has a better performance per cost when compared to the digital hardware accelerators. In Table 6 we also compare the area of various processors reported in literature and we find that the analog co-processor reported in this work has the best performance per cost among these processors.

4.5. Fabrication of Microscale Memristor Devices and Device Characterization

One of the key tasks in this project was to fabricate and characterize microscale memristor devices. A set of microscale memristor devices were fabricated at UMass. Figure 53 shows one layer of the mask that was used for the device fabrication. The mask contains twelve rows, with each row containing 16 individual microscale memristor devices. The mask consists of five layers. Layer 1 which is shown in the figure acts as the bottom electrode and the foundation for the device. Layer 2 is the device active area which contains the switching material. Layer 3 is used to cover the active area to prevent any damage during the etching process. Layer 4 is the top electrode and layer 5 is the passivation opening for the pads.

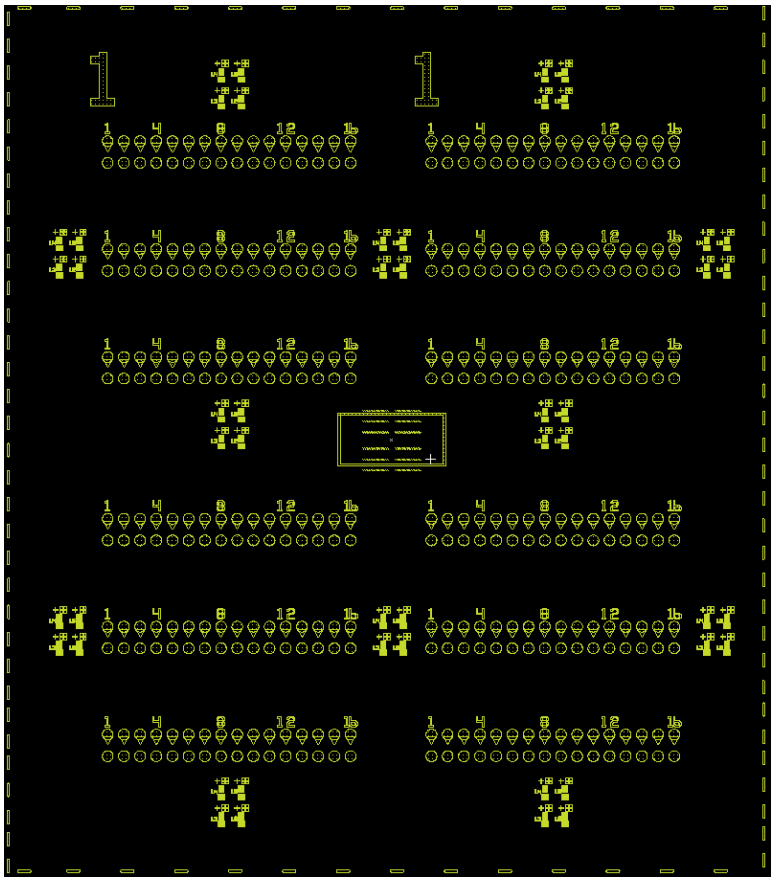


Figure 53: Layer 1 of the Mask used for microscale device fabrication

4.5.1. The Device Fabrication Process

The microscale memristor devices were fabricated on a substrate of Si wafers that have 100 nm thermally grown SiO_2. The feature size of the memristor devices is 10 × 10 μm^2. The bottom electrodes were patterned by ultraviolet photolithography. After that, 1.5 nm Ti and 20 nm Pt were deposited sequentially by electron beam evaporator, followed by a lift-off process in acetone. A 5 nm HfO_2 blanket layer was prepared by atomic layer deposition ("ALD") using water and tetrakis (dimethylamido) hafnium as precursors at 250° C. 50 nm thick Ta top electrodes were defined by a second photolithography step and a 15 s O_2 descum, metallization using DC sputtering and liftoff.

4.5.2. Device Characterization and Measurements

After the device is fabricated and before the device can be operational, a large voltage is applied. This voltage is called the forming voltage and the process is called forming. This creates conduction channels or filaments between the two electrodes of the memristor device. Figure 54 shows the response of one of the devices for an applied forming voltage of 4.2 V. Once the devices were formed they were set using a voltage of 1.4 V and a compliance current of 10 µA as shown in Figure 55. During the set process, more filaments are formed and the memristor device moves to a high conductance state. The devices were then reset using a voltage of -2 V as shown in Figure 56. During the reset process the conduction filaments are broken and the device moves to a low conductance state.

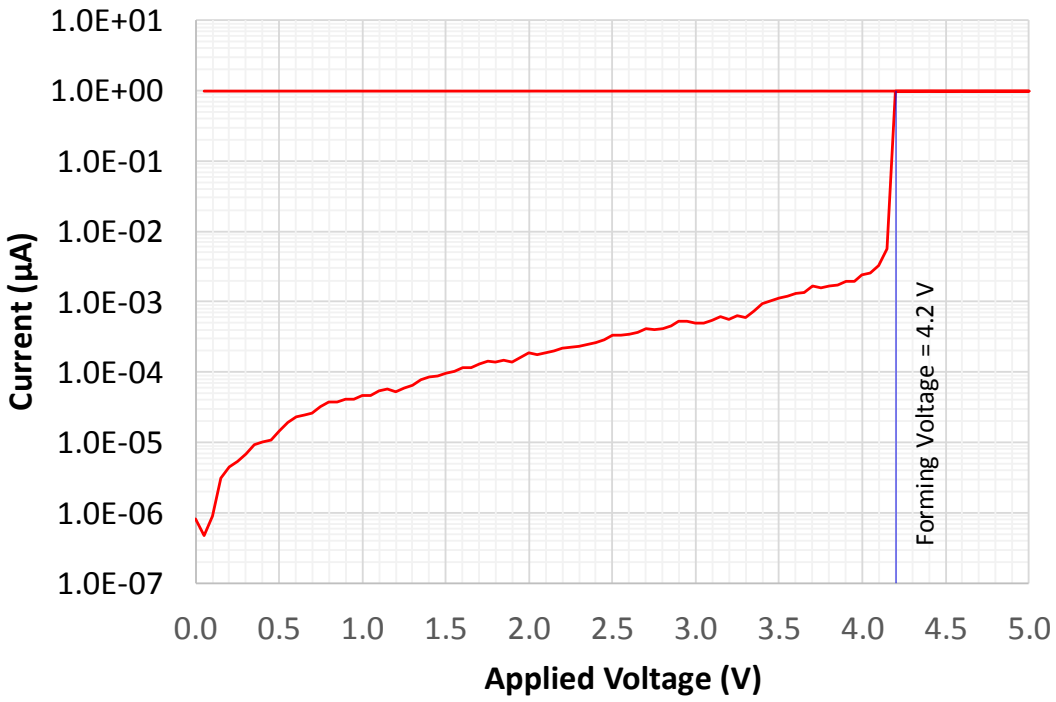


Figure 54: Forming voltage of the memristor device

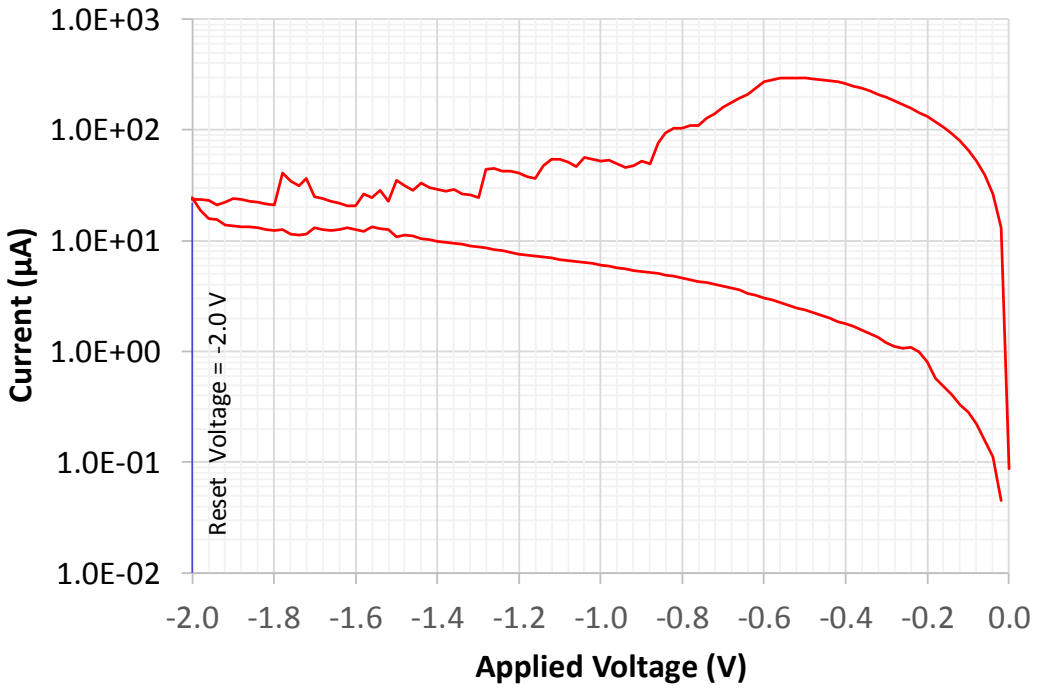


Figure 55: Reset voltage of the memristor device

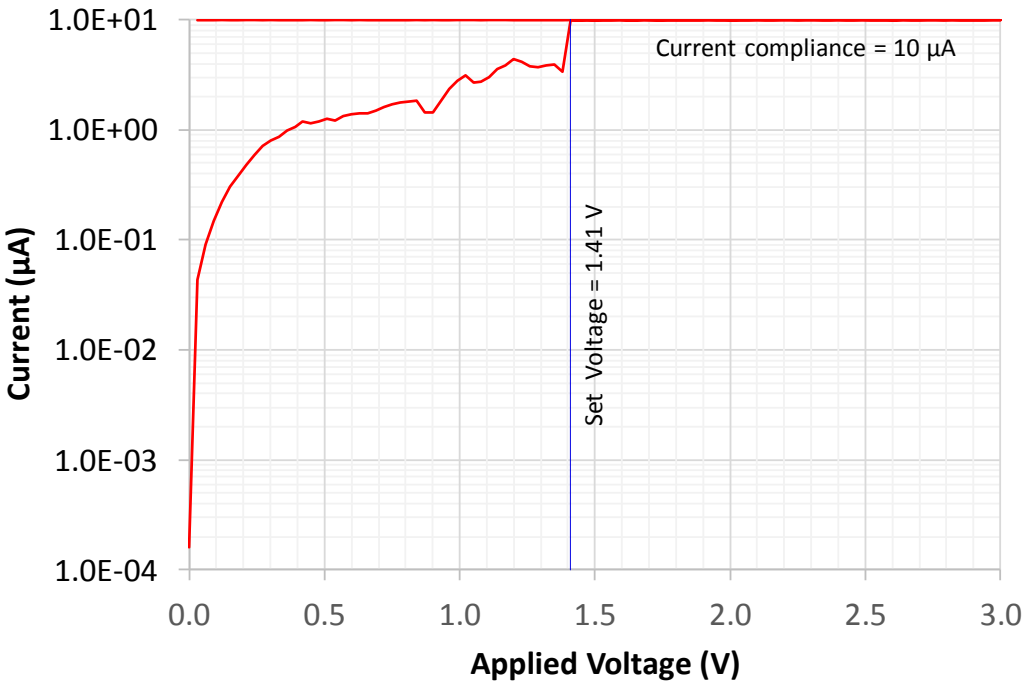


Figure 56: Set voltage of the memristor device

After the devices were fabricated and initial measurements were performed, the devices were to be characterized to measure s-parameters, conductance variations, read speeds and

number of bit per cell. In order to perform these measurements very accurately, the measurement equipment needs to be de-embedded up to the Device Under Test ("DUT"). Hence, we decided to build a memristor device characterization board that can be de-embedded very easily and can be used for very accurate and precise device characterization.

4.5.3. Characterization Board to Evaluate Microscale Memristor Devices

Figure 57 shows a block diagram of the characterization board that is being developed to characterize the newly fabricated memristor devices. The board has two channels, one for each memristor device being characterized. Each channel has an input SMA connector followed by a 5dB attenuator to attenuate any reflections coming from the device because of impedance mismatches. The attenuator is followed by the DUT which is the memristor device. Following the DUT is a resistor. The DUT and the resistor together act as a voltage divider network. The other end of the resistor is connected to an output SMA connector. All the devices are connected through a 50 Ω backed coplanar waveguide structure designed on the PCB.

As mentioned earlier, it is very important to de-embed the board up to the DUT to perform accurate measurements on the device and characterize them. To enable this, we have a separate de-embed board with 'short' and 'load' structures as shown in the block diagram of Figure 58. To de-embed the board for the 'open' case, the board in Figure 57 will be used without populating the DUT. Figure 59 shows the PCB layout of the characterization board and Figure 60 shows the PCB layout of the de-embedding board.

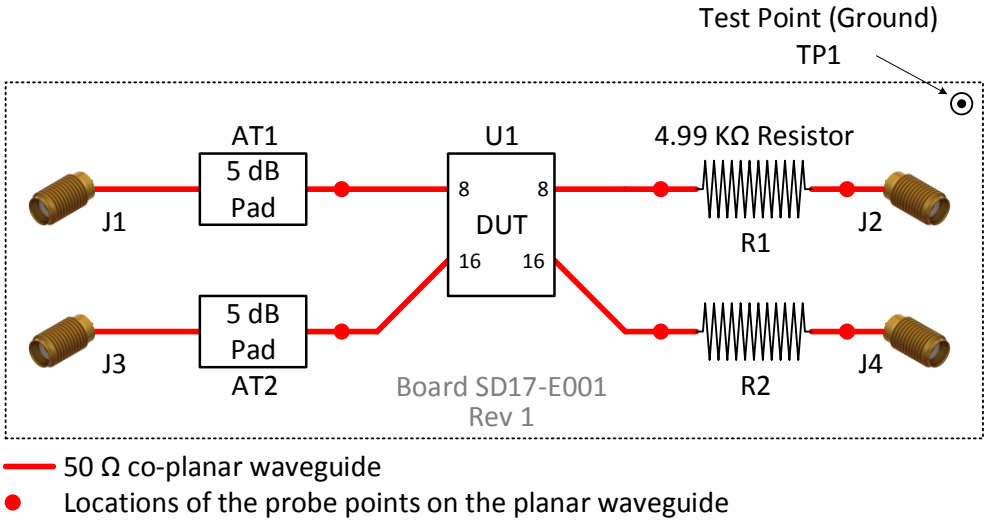


Figure 57: Block diagram of the characterization board

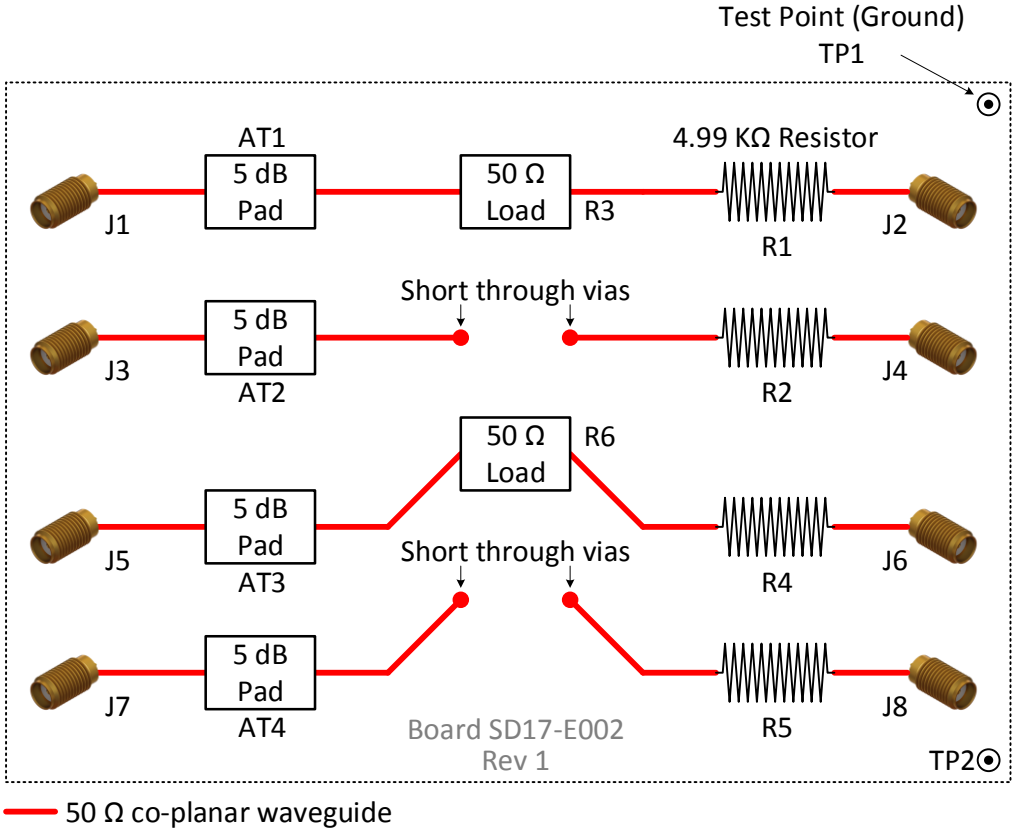


Figure 58: Block diagram of the de-embedding board

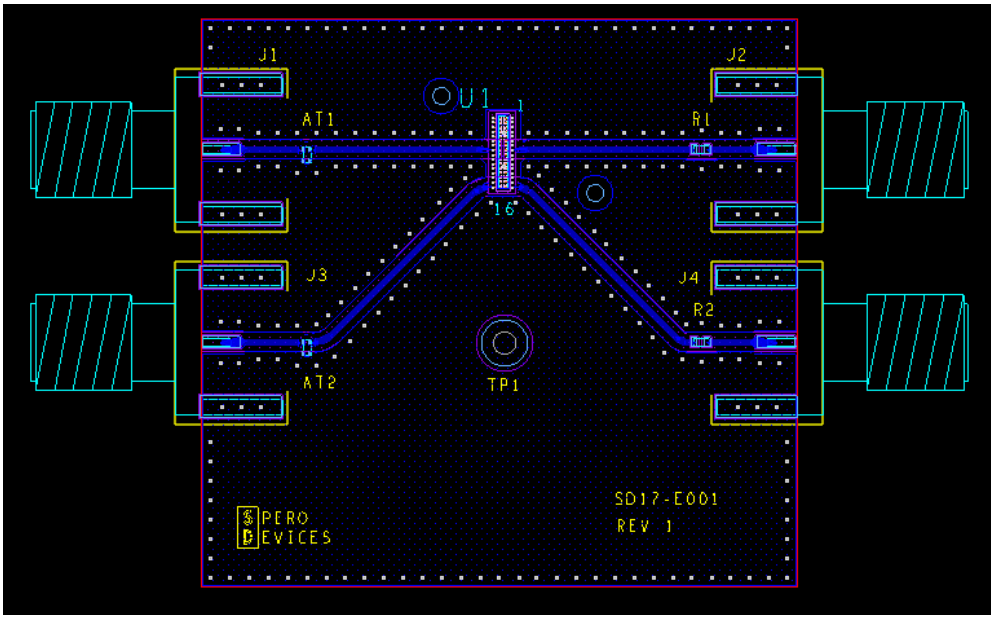


Figure 59: PCB Layout of the characterization board

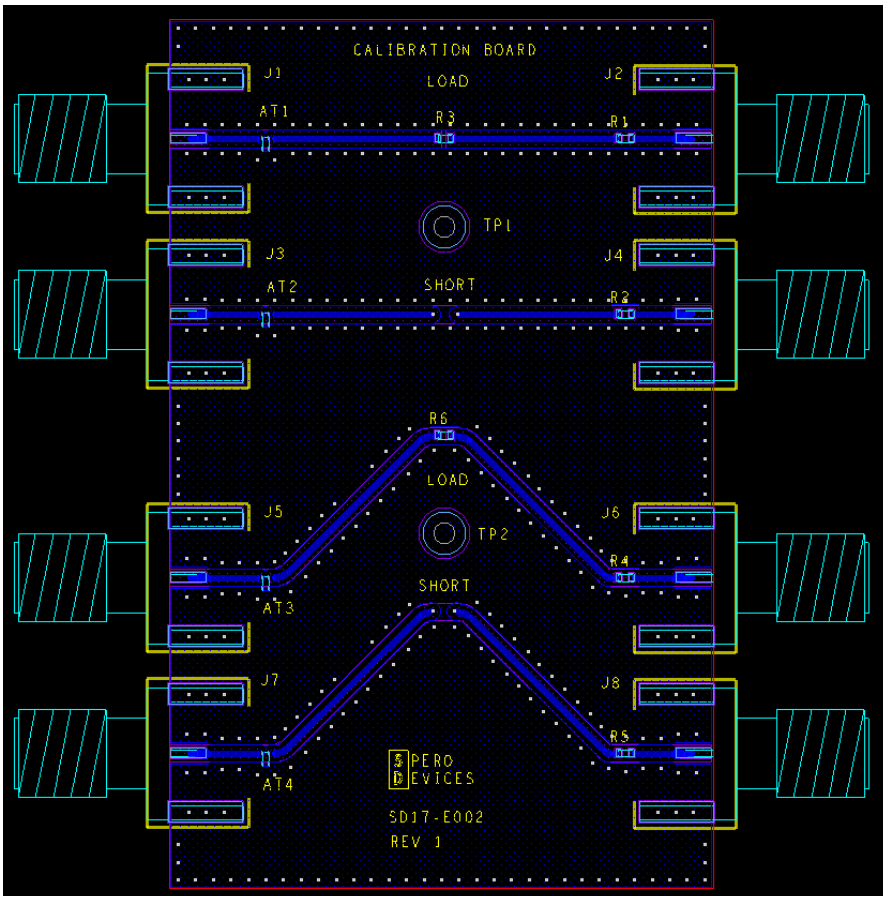


Figure 60: PCB Layout of the de-embedding board

Figure 61 shows the three memristor tiles that were fabricated at UMass and Figure 62 shows the magnified image of one of the tiles with the top and bottom electrode marked.

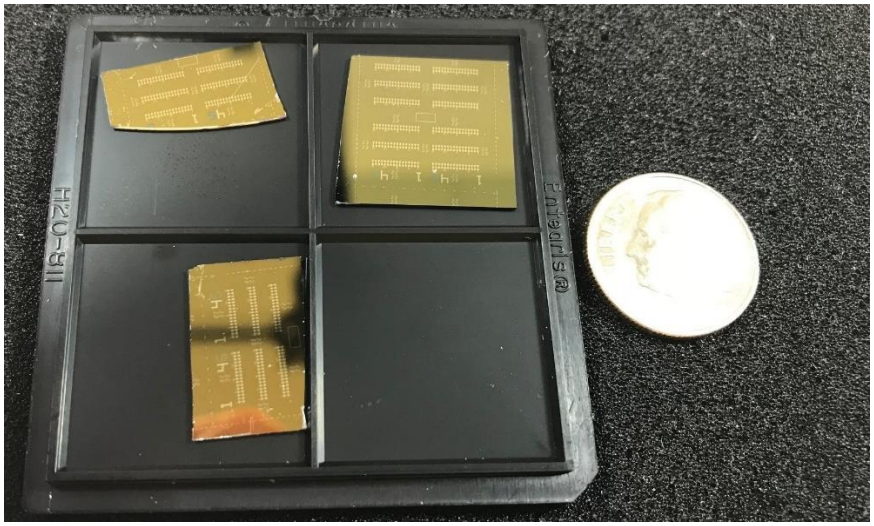


Figure 61: Three tiles that were fabricated

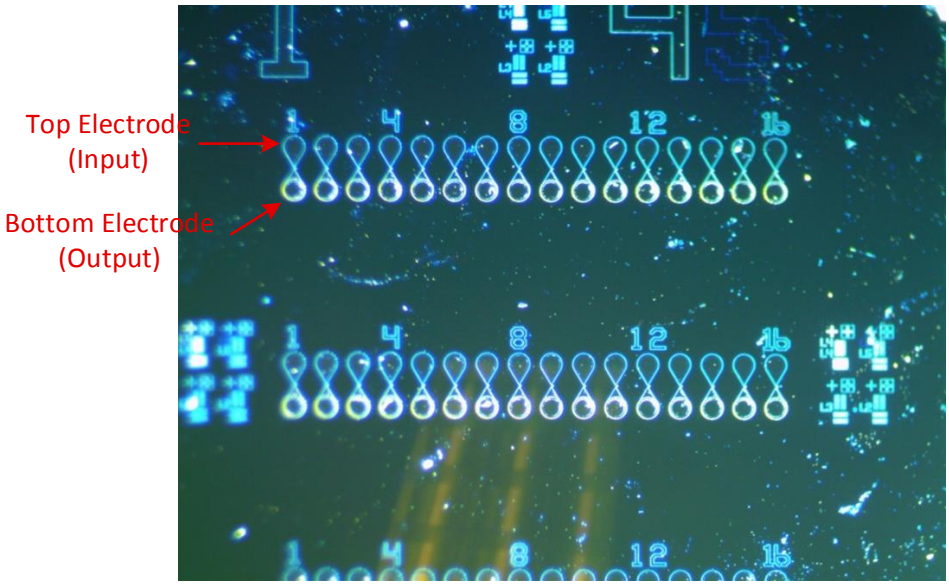


Figure 62: Magnified image of two 1 x 16 arrays on the tile

Figure 63 and

Figure **64** show the populated characterization boards and de-embedding boards. The top straight structure in Figure 63 will henceforth be referred to as 'Structure 1' or 'first structure' and

the bottom angled structure in Figure 63 will henceforth be referred to as ‘Structure 2’ or ‘second structure’.

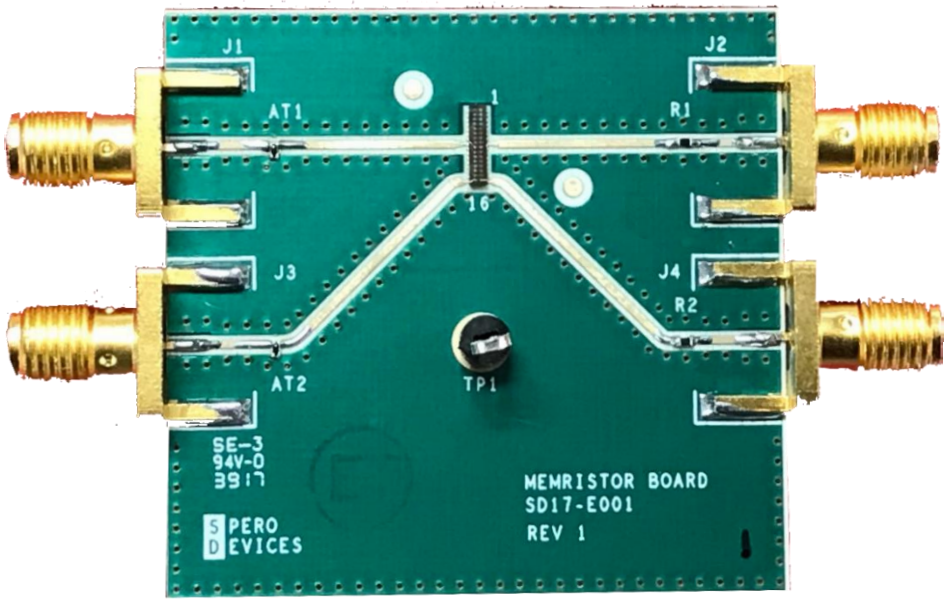


Figure 63: Image of the fabricated characterization boards

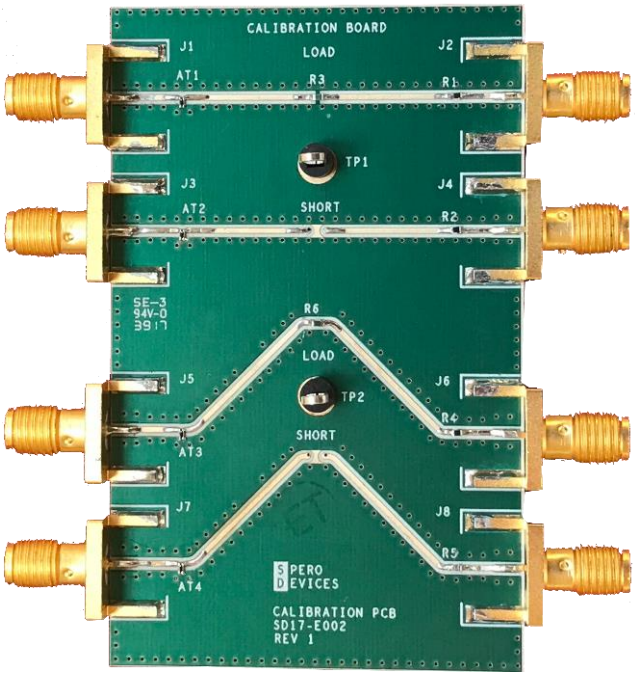


Figure 64: Image of the fabricated de-embedding boards

4.5.4. Measurement and Characterization of Microscale Memristor Devices

4.5.4.1. De-embedding the Memristor Characterization Board

The large variety of printed circuit transmission lines makes it difficult to create test equipment that can easily interface to all the different types and dimensions of microstrip and coplanar transmission lines. The test equipment requires an interface to the selected transmission media through a test fixture. Accurate characterization of the device under test ("DUT") requires the test fixture characteristics to be removed from the measured results. The test equipment typically used for characterizing the RF and microwave component is the vector network analyzer ("VNA") which uses standard 50 Ω or 75 Ω coaxial interfaces at the test ports. The test equipment is calibrated at the coaxial interface defined as the "measurement plane," and the required measurements are at the point where the DUT attaches to the printed circuit board, or the "device plane" as shown in Figure 65. When the VNA is calibrated at the coaxial interface using any standard calibration kit, the DUT measurements include the test fixture effects.

Over the years, many different approaches have been developed for removing the effects of the test fixture from the measurement. De-embedding uses a model of the test fixture and mathematically removes the fixture characteristics from the overall measurement. This fixture "de-embedding" procedure can produce very accurate results for the non-coaxial DUT, without complex non-coaxial calibration standards. The process of de-embedding a test fixture from the DUT measurement can be performed using scattering parameter matrices.

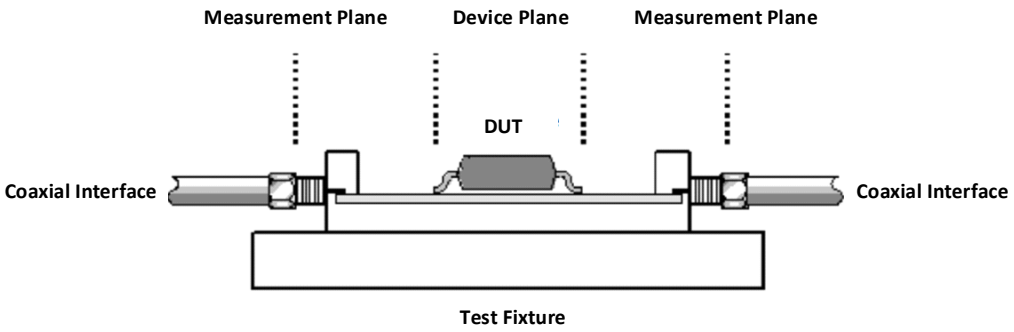


Figure 65: Test fixture configuration showing the measurement and device planes

4.5.4.2. De-embedding Procedure

We have built three test structures to perform the de-embedding of the microscale characterization board. The de-embedding technique that we use here is the Short-Open-Load technique.

1. 'Open' De-Embedding Procedure (One Step De-Embedding Method)

   The most dominant parasitics are in parallel with the DUT and can be modeled as capacitance with resistance. In this method, two s-parameter measurements are done – one with an open test structure and the other with the test structure plus DUT. These parameters are converted to y-parameters. The y-parameters of the open test struycture is subtracted from the y-parameters of the test structure plus DUT to obtain y-parameters of the DUT.

   1. Measure s-parameters of open test structure [S_O] and s-parameters of test structure plus DUT [S_T].
   2. Convert S-parameters to Y-Parameters: [Y_O] and [Y_T]
   3. Subtract: [Y_{DUT}] = [Y_T] – [Y_O]
   4. Convert: Y_{DUT} to S_{DUT}

2. ‘Short’ De-Embedding Procedure (One Step De-Embedding Method)

   The most dominant parasitics are in series with the DUT. In this method, two s-parameter measurements are done - one with a short test structure and the other with the test structure plus DUT. These s-parameters are converted to z-parameters. The z-parameters of the short test structure is subtracted from the z-parameters of the test structure plus DUT to obtain z-parameters of the DUT.

   1. Measure S-Parameters of open test structure [S_O] and S-parameters of test structure plus DUT [S_T].
   2. Convert S-parameters to Z-Parameters: [Z_O] and [Z_T]
   3. Subtract: [Z_{DUT}] = [Z_T] – [Z_O]
   4. Convert: Z_{DUT} to S_{DUT}

3. ‘Load’ De-Embedding Procedure (One Step De-Embedding Method)

   1. Measure S-Parameters of open test structure [S_L] and S-parameters of test structure plus DUT [S_T].
   2. Convert S-parameters to Z-Parameters: [Z_L] and [Z_T]
   3. Subtract: [Z_{DUT}] = [Z_T] – [Z_L]
   4. Convert: Z_{DUT} to S_{DUT}

4.5.4.3. Trial Measurements Procedure

Trial measurements were initially performed on the memristor devices. The purpose of the trial measurements was to obtain approximate values of the operating voltage and current of the fabricated microscale memristor device. These measurements were performed using the B1500 Semiconductor Parameter Analyzer. The procedures for the trial measurement are listed below.

1. A small voltage (e.g. 2 V) is applied to the device with a fixed compliance current (e.g. 1 μA) to determine the switching voltage of the device. The voltage and compliance currents are increased gradually until the devices switches.
2. The memristor device is read again to ensure that the switching is non-volatile. If it is not non-volatile, the voltage is fixed, and the compliance current is increased until the switching becomes non-volatile. This process is referred as 'forming the device.'
3. The device is then reset without setting a compliance current. The reset voltage is gradually increased to achieve high ON/OFF ratio. This technique is used to achieve the highest ON/OFF ratio. From this approach, we obtain the upper bound of the memristor conductance values G_{MAX} and the lower bound of the memristor conductance value G_{MIN}.
4. The memristor devices sometimes work better with different polarities of voltage. Steps (a) to (c) are repeated for a negative forming/SET voltages and positive RESET voltages. If they result in more stable switching behavior and large ON/OFF ratio, then this approach will be used.

4.5.4.4. Read Speed Measurements Procedure

The goal of this measurement is to find the read speed of the microscale memristor devices. The speed of the VMM operation directly depends on the read speed of these devices. The measurement procedure is listed below.

1. The microscale memristor device is directly probed using the probe station shown in and the conductance is measured using the by Keysight B1500 semiconductor analyzer.
2. Low parasitic cables are used to connect a function generator (Keysight 33250) to the characterization board.

3. The function generator is used to send read voltage pulses to the device and the voltage drop across the 4.99 KΩ resistor is measured. This voltage drop is used to determine the conductance values of the memristor device.
4. The read voltage pulse widths are gradually reduced till the reading is not precise. The lowest usable pulse width determines the read speed of the memristor devices.

4.5.4.5. Multiple Bits Per Cell

The goal of this measurement is to demonstrate that the memristor device can achieve multiple resistance/conductance levels and that these levels are linear. The test procedure for this measurement is as follows:

1. A maximum (G_{MAX}) and minimum (G_{MIN}) conductance range is first chosen for the memristor.
2. The memristor device is set to a conductance state between G_{MAX} and G_{MIN} using a set voltage and a compliance current. To set the memristor to different conductance states, the preferred approach is to change the compliance current. If this approach does not work, then the set voltage can be varied to achieve different conductance states.
3. After setting the memristor to a conductance state the device is read multiple times using read voltage pulses.
4. Steps (b) and (c) are repeated till 32 levels are demonstrated on the memristor device.

4.5.4.6. Read Variation

The goal of this measurement is to get statistical data on the read stability of microscale memristor devices. The test procedure for this is to repeat the steps listed in section 4.5.4.5 for multiple memristor devices.

4.5.5. Test Results for Microscale Memristor Characterization

We measure the s-parameters of various structures on the de-embedding board and the characterization board using a VNA. The s-parameters are measured from 10 MHz to 1 GHz and they are used in the de-embedding procedure as mentioned in section 4.5.4.2. Figure 66 to Figure 72 are explained below.

1. Figure 66 shows the 2-port s-parameters of the co-planar waveguide on the de-embedding board. This is done by replacing the attenuator, 50 Ω load and the 4.99 KΩ resistor with 0 Ω resistors. The s-parameters for the transmission line measured at 250 MHz are s_{11} = -35.4 dB, s_{12} = -0.0584 dB, s_{21} = 0.031 dB and s_{22} = -35.8 dB.

2. Figure 67 shows the 2-port s-parameters of the 50 Ω load structure on the de-embedding board which includes the 5 dB attenuator and 4.99 KΩ resistor, but with the 50 Ω replaced with 0 Ω. The s-parameters measured at 250 MHz are s_{11} = -9.47 dB, s_{12} = -55.6 dB, s_{21} = -55.4 dB and s_{22} = -0.177 dB. The return loss is around 10 dB as expected because of the 5 dB attenuator. The insertion loss is high because of the 4.99 KΩ resistor.

3. Figure 68 shows the 2-port s-parameters of the 50 Ω load structure on the de-embedding board which includes the 5dB attenuator and 4.99 KΩ resistor, both of which are replaced with 0 Ω. The s-parameters measured at 250 MHz are s_{11} = -9.51 dB, s_{12} = -3.58 dB, s_{21} = -3.59 dB and s_{22} = -9.44 dB. The return loss is about -10 dB, as expected, because of the 5 dB attenuator. The insertion loss is about -3.5dB because of the 50 Ω load resistor.

4. Figure 69 shows the S_{11} values of five different memristors on the first structure of the memristor characterization board set to different conductance states. The five S_{11}

values measured at 250 MHz are -12.7 dB (‘red’), -13.8 dB (‘blue’), -12.3 dB (‘green’), -12.5 dB (‘orange’) and -11.7 dB (‘purple’). These values are the return loss of the memristors at different conductance values.

5. Figure 70 shows the S_{21} values of five different memristors on the first structure of the memristor characterization board set to different conductance states. The five S_{21} values measured at 250 MHz are -44.9 dB (‘red’), -43.7 dB (‘blue’), -47.8 dB (‘green’), -45.1 dB (‘orange’) and -49.4 dB (‘purple’). These values are the insertion loss of the memristors at different conductance values.

6. Figure 71 shows the S_{11} values of five different memristors on the second structure of the memristor characterization board set to different conductance states. The five S_{11} values measured at 250 MHz are -12.5 dB (‘red’), -11.6 dB (‘blue’), -12.1 dB (‘green’), -11.2 dB (‘orange’) and -12.8 dB(‘purple’). These values are the return loss of the memristors at different conductance values.

7. Figure 72 shows the S_{21} values of five different memristors on the first structure of the memristor characterization board set to different conductance states. The five S_{21} values measured at 250 MHz are -48.1 dB (‘red’), -51.8 dB (‘blue’), -49.7 dB (‘green’), -55.5 dB (‘orange’) and -44.1 dB (‘purple’). These values are the insertion loss of the memristors at different conductance values.

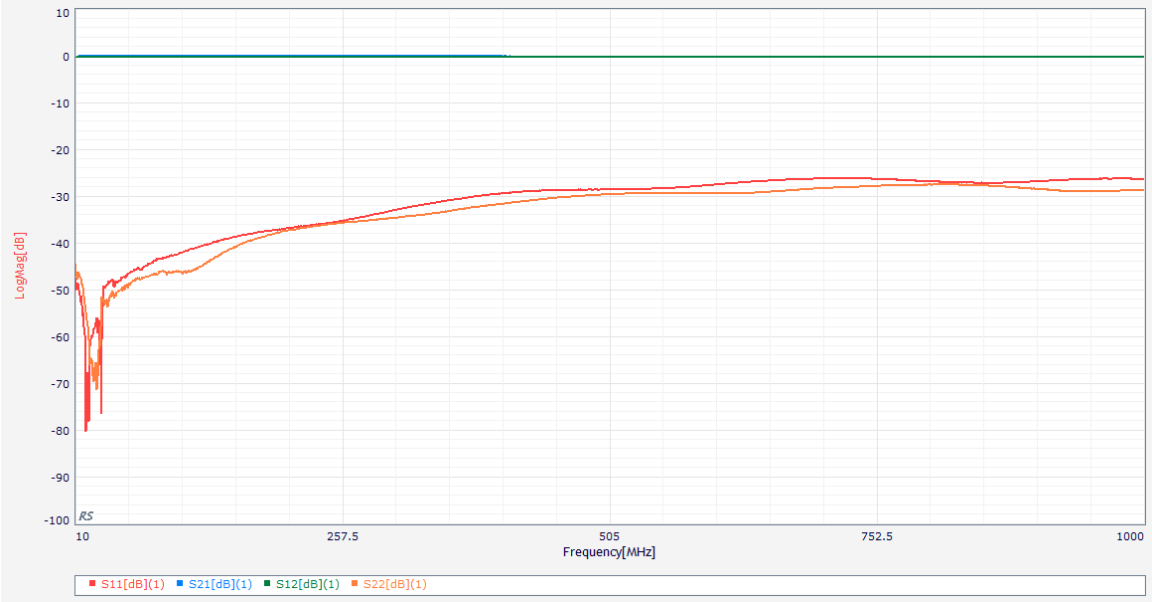


Figure 66: S-parameters of the transmission line

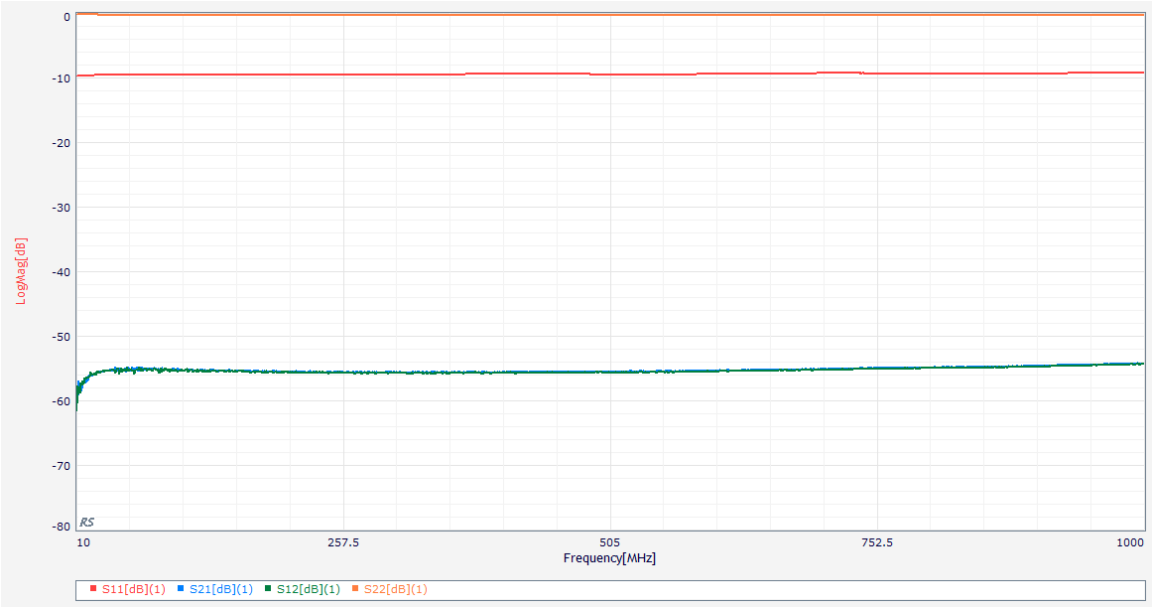


Figure 67: S-Parameters of first structure on the de-embedding board with the 5dB attenuator and 4.99 KΩ resistor

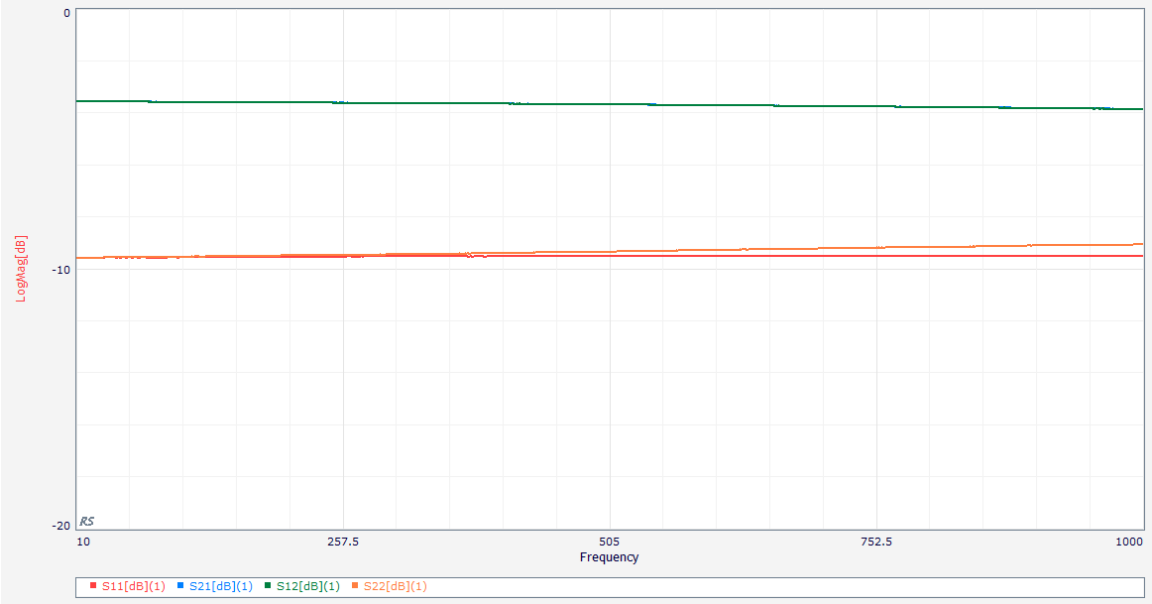


Figure 68: S-Parameters of first structure on the de-embedding board with the 5dB attenuator and 4.99 KΩ resistor replaced with 0Ω

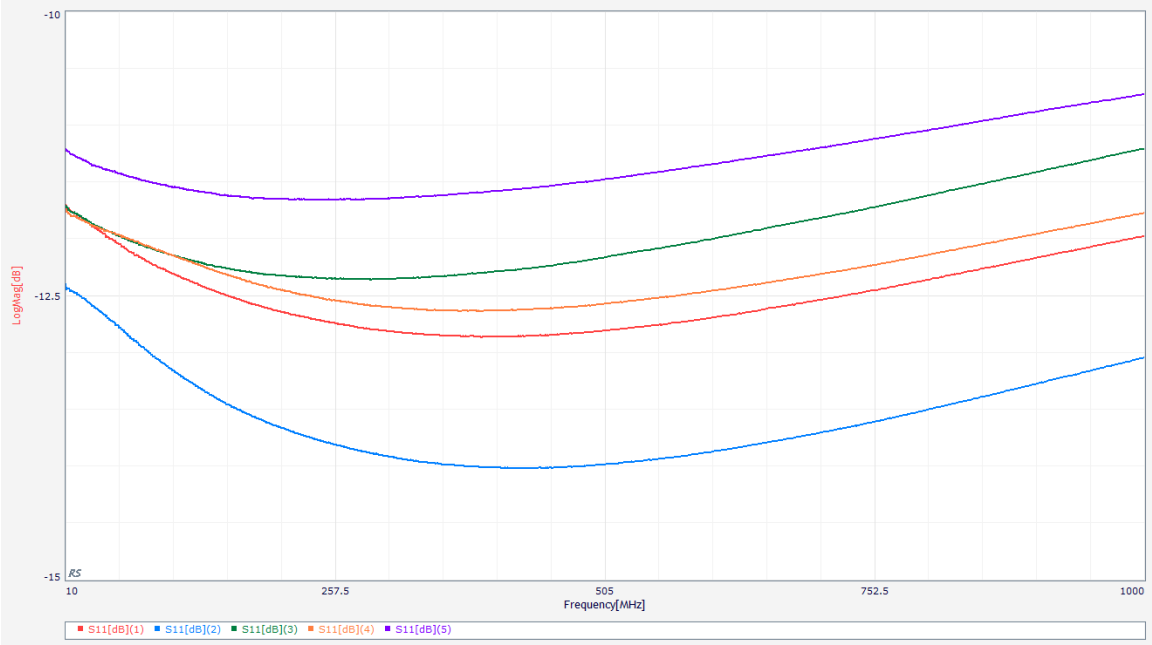


Figure 69: S_{11} of five memristor devices on the first structure of the memristor characterization board

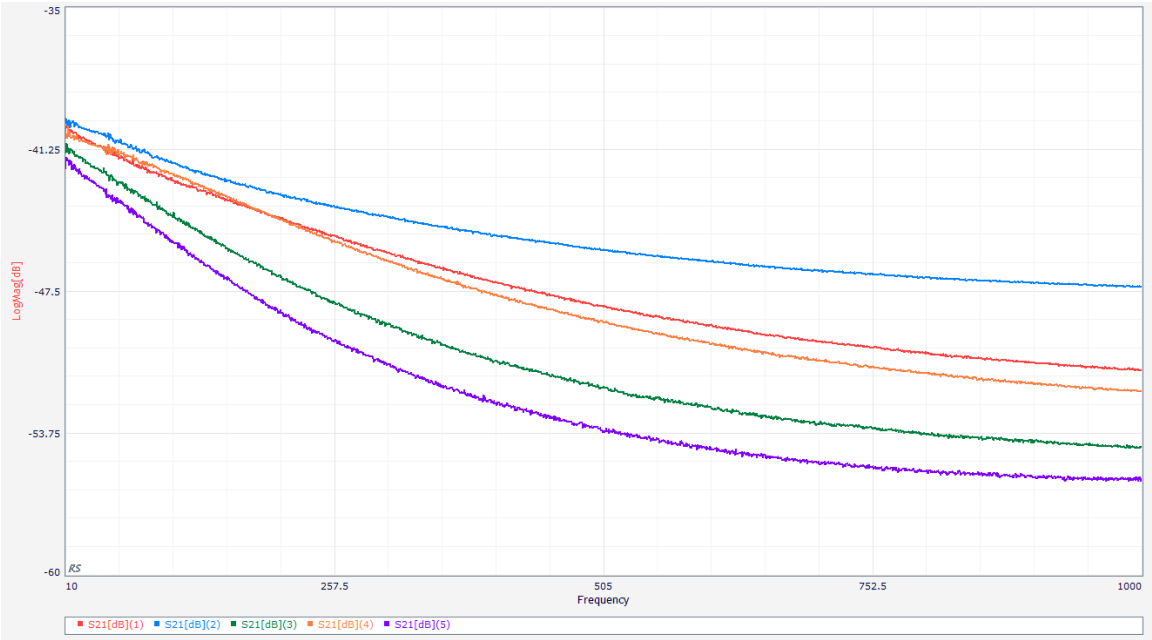


Figure 70: S_{21} of five memristor devices on the first structure of the memristor characterization board

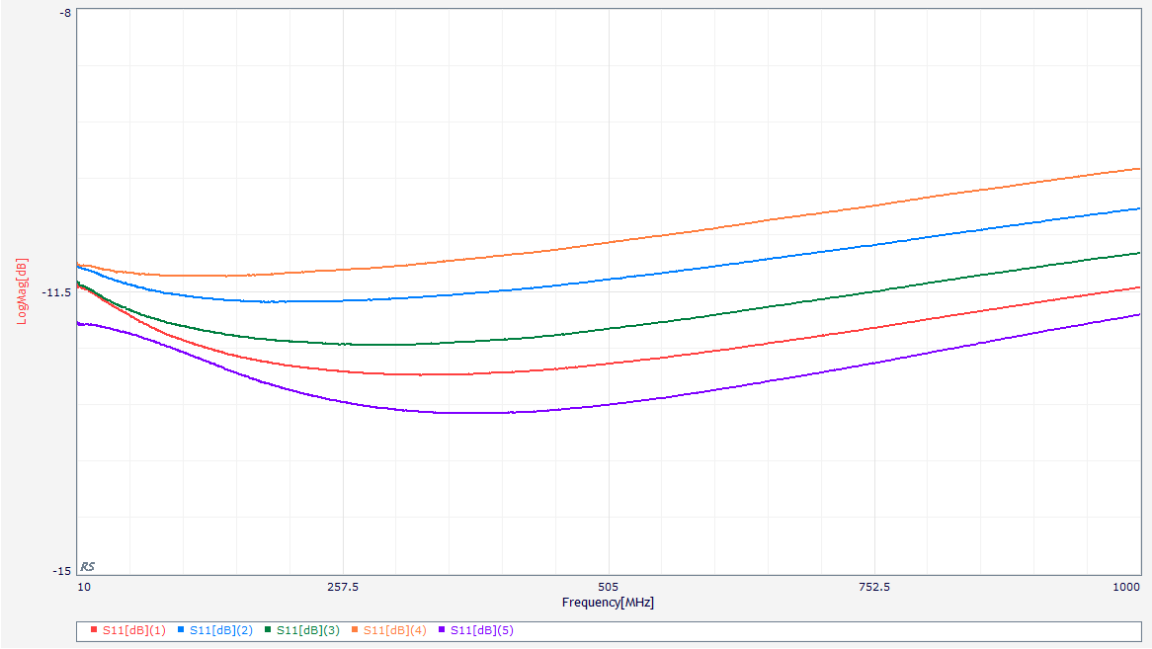


Figure 71: S_{11} of five memristor devices on the second structure of the memristor characterization board

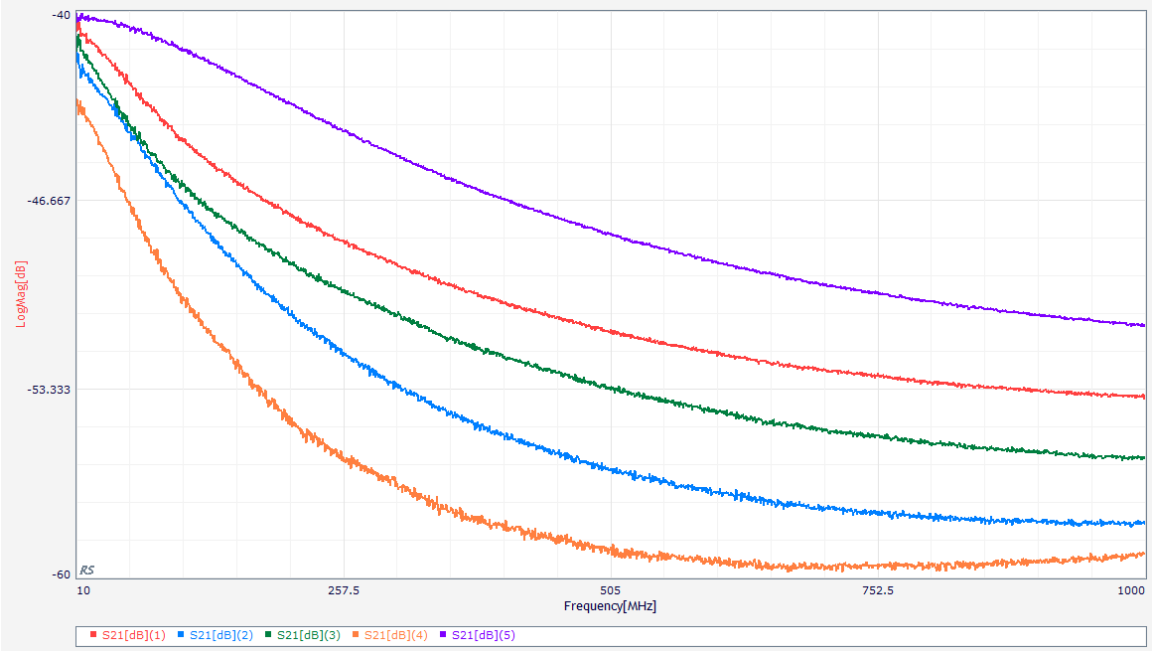


Figure 72: S_{21} of five memristor devices on the second structure of the memristor characterization board

4.5.6. De-embedding using ADS

The measured S-parameter values can be used to de-embed the board using ADS tool. The first step is to build a parasitic model of the memristor device. Figure 73 shows the model being considered. The memristor resistance is represented by a DC resistance value 'Rdc'. The memristor has an internal inductance value represented by L1. It has an internal shunt capacitance represented by Cs. As the memristor is placed on a PCB, Cp represents the parasitic capacitance to ground.

As there are other components on the characterization board, they have to be de-embedded. One method that can be used to do this is to construct a model of the board and its components in ADS and then simulate its S-parameters and compare it to the S-Parameters measurements made using the VNA. The parasitic values of the memristor model are tweaked within reasonable limits till the simulated value is close to the measured values.

Figure 74 shows the model of the memristor characterization board developed in ADS. It has two 50 Ω terminations at either end followed by a coplanar waveguide structure which is also designed for 50 Ω. The memristor equivalent circuit model which is the DUT is at the center. The memristor device is wire bonded to a 5dB attenuator on the left and to a 5 KΩ resistor on the right. This is also modeled as shown in the figure.

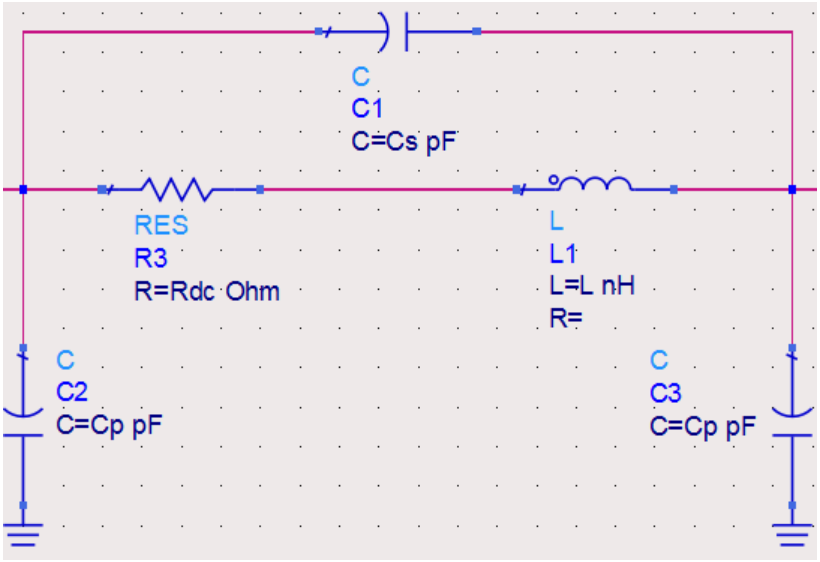


Figure 73: Parasitic/Equivalent Circuit model of the microscale memristor device

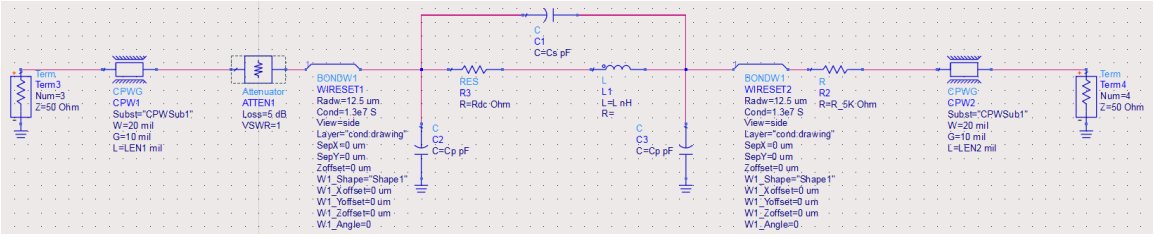


Figure 74: Model of the microscale memristor characterization board

Using this model, we compare the simulated S-parameters to measured S-parameters for eleven devices. The values being compared are log magnitudes of S_{11} and S_{12} and phase of S_{12}. Figure 75 to Figure 85 shows the comparison of simulated and measured S-parameters for eleven different devices. In these figures, ports 1 and 2 are used for measured S-parameter values and ports 3 and 4 are used for simulated S-parameters.

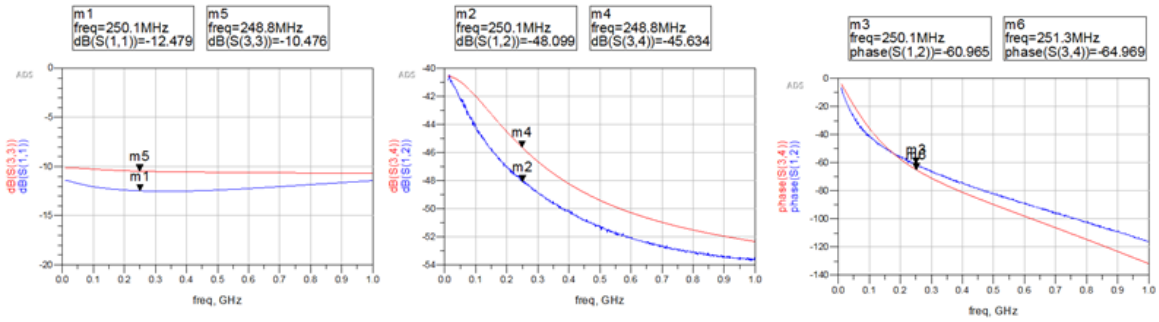


Figure 75: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 1

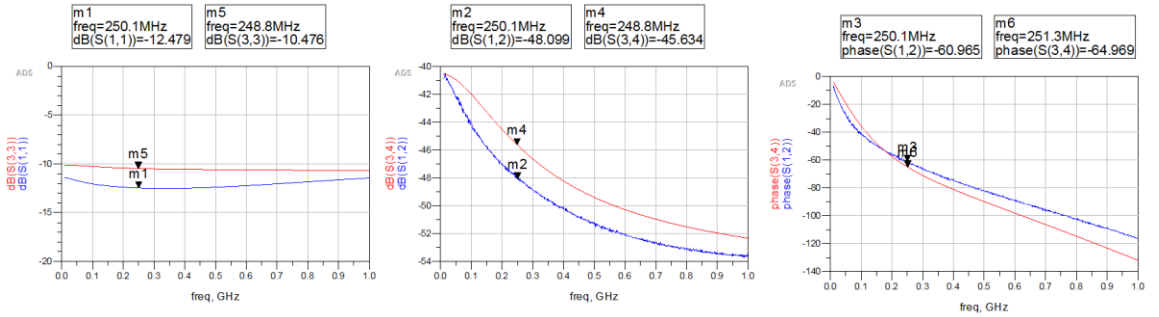


Figure 76: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 2

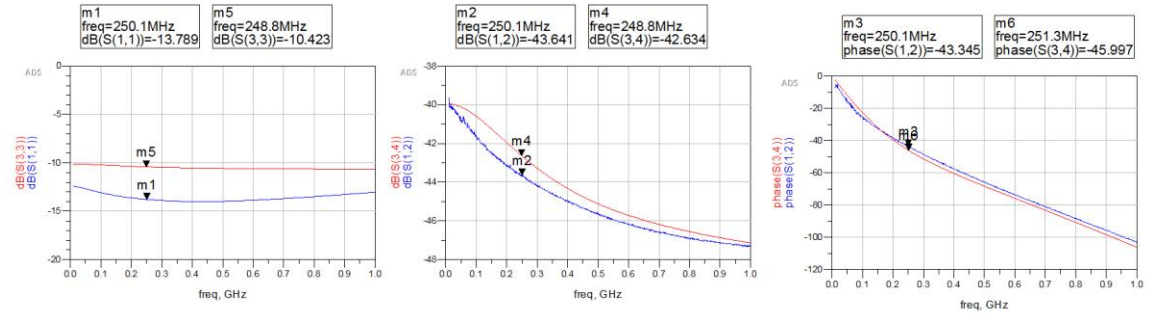


Figure 77: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 3

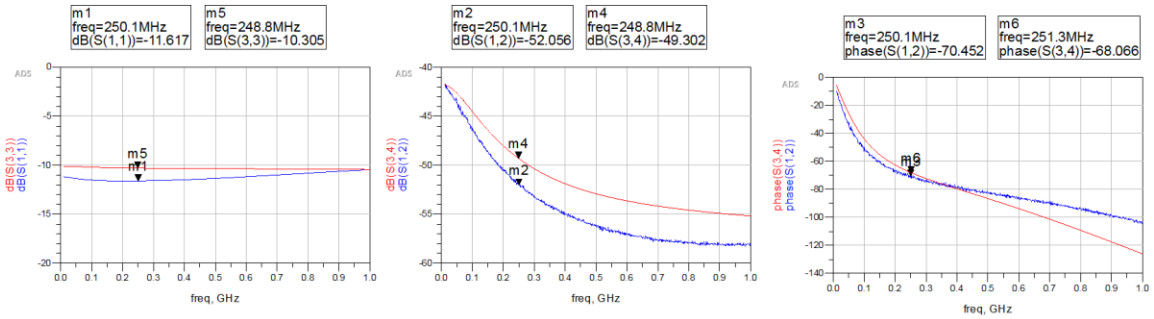


Figure 78: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 4

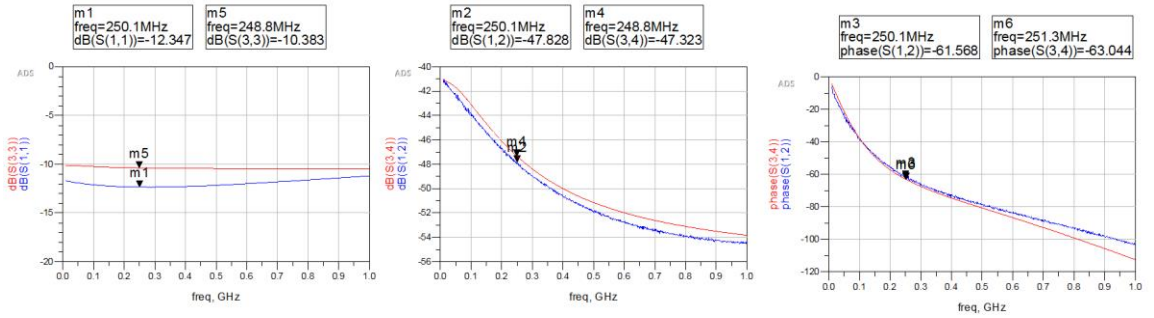


Figure 79: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 5

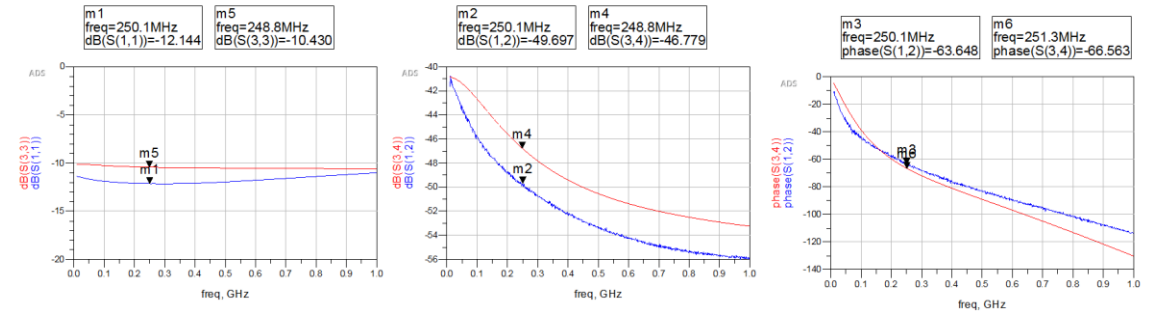


Figure 80: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 6

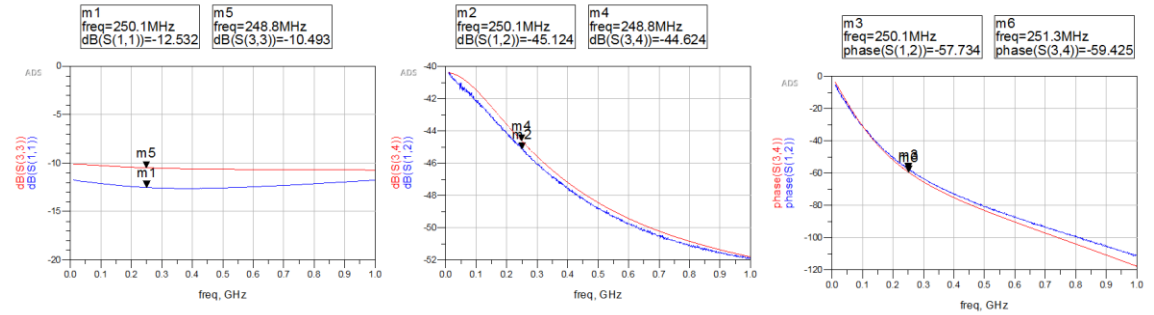


Figure 81: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 7

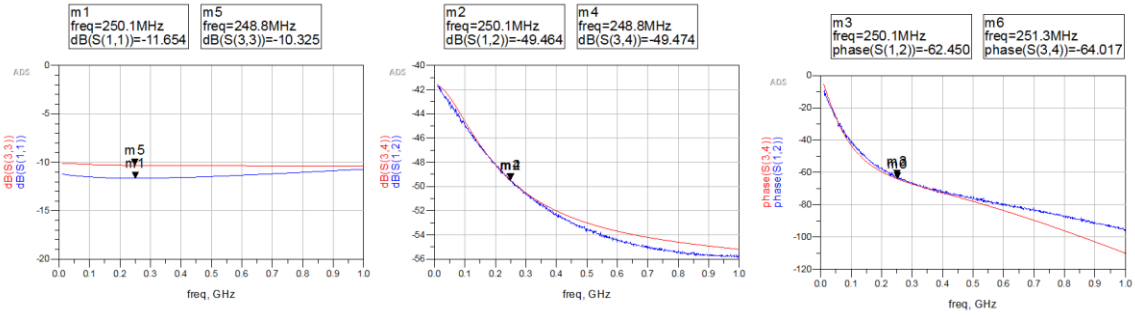


Figure 82: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 8

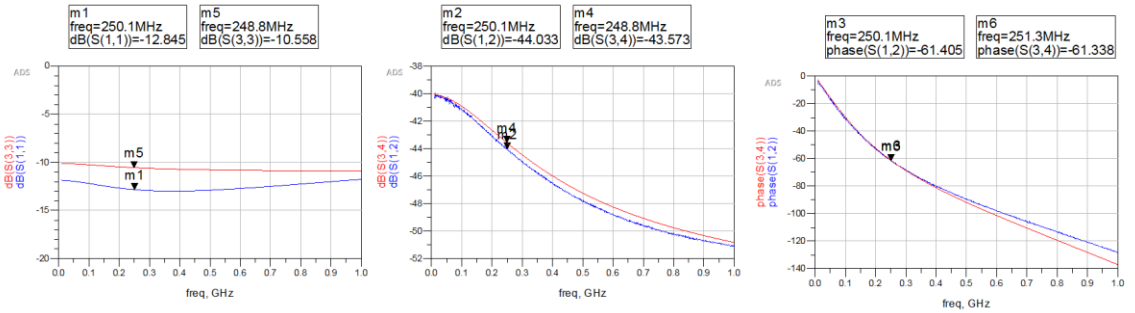


Figure 83: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 9

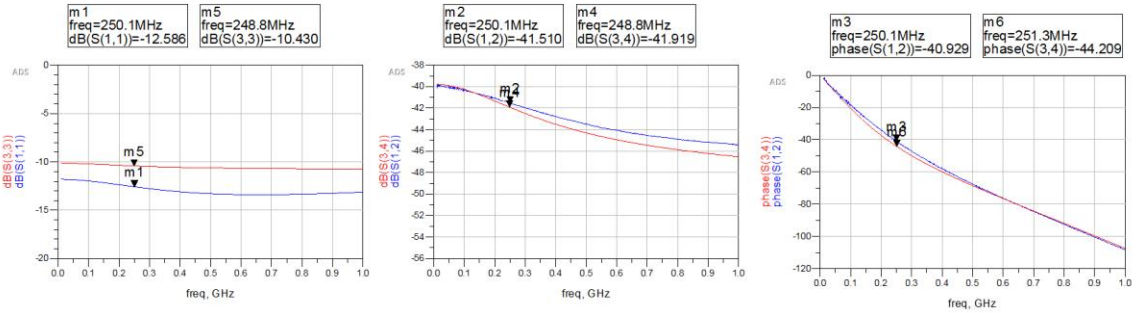


Figure 84: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 10

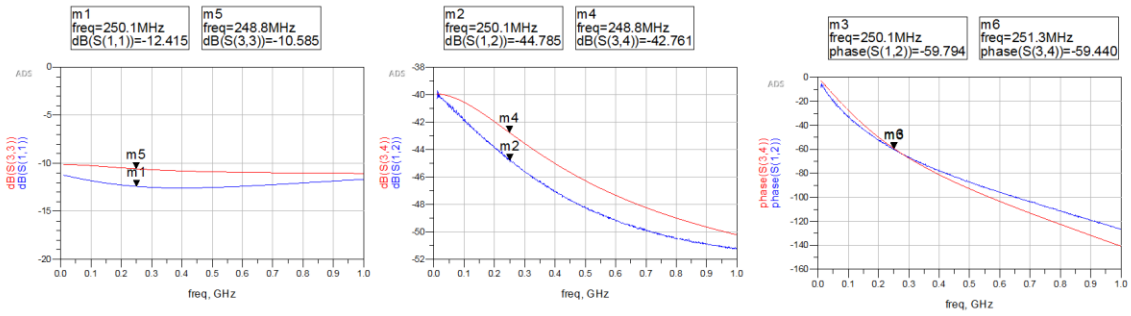


Figure 85: S-parameter comparison between measured data ('blue') and simulated data ('red') for device 11

Table 7 shows the resistance, capacitance and inductance values for each device to match the simulated and measured S-parameters. There are two structures on the memristor characterization board. The top straight structure is labeled as 'structure 1' and the angled bottom structure is labeled 'structure 2'. The 'Rdc' values for each of these devices were set to a different value and those values are observed here. The width of the chip after the wire bond is around 70 mils. The inductance was found to be around 1.5nH for this width. The shunt capacitance Cs and the external capacitance to ground C_p are varied to match the simulated and measured S-parameters. The values of C_s are very close to each other except for device 3 and device 10. If these values are considered outliers and eliminated, then the average value of Cs is 0.246pF. The values of C_p are all close to each other and the average of all the eleven devices is 1.02pF.

Table 7: Comparison of the proposed work with other processors reported in literature

Device Number	Board Structure	Rdc (Ω)	L (nH)	Cs (pF)	Cp (pF)
1	Structure 1	580	1.5	0.24	0.99
2	Structure 2	880	1.5	0.27	1.07
3	Structure 1	480	1.5	0.64	0.95
4	Structure 2	1780	1.5	0.18	0.99
5	Structure 1	1280	1.5	0.21	1.00
6	Structure 2	1080	1.5	0.25	1.08
7	Structure 1	780	1.5	0.25	1.00
8	Structure 1	1680	1.5	0.20	1.09

9	Structure 2	580	1.5	0.31	1.07
10	Structure 1	380	1.5	0.70	0.95
11	Structure 2	480	1.5	0.31	1.07
Average values			1.5	0.246	1.02

Figure 86 shows the parasitic model of the microscale memristor device which uses the average C_s and C_p values.

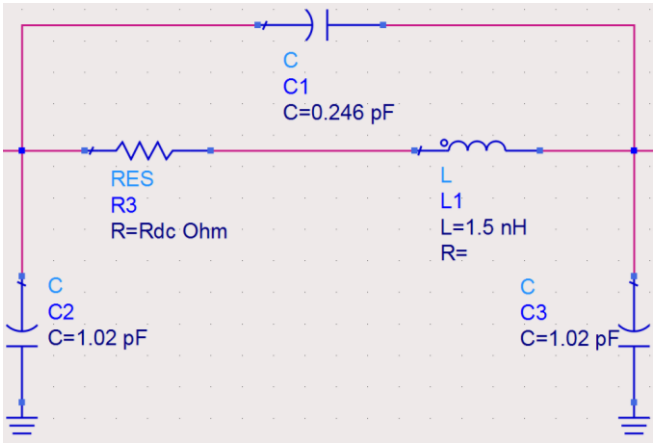


Figure 86: Parasitic model of the microscale memristor device obtained from ADS

This parasitic memristor model can now be used in any of our future analog co-processor designs.

4.5.7. Implementing Trial Measurements

The test procedure for the trial measurement is described in section 4.5.4.3. Figure 87 shows the I-V curves for five memristor devices during the forming process. The compliance current is set at 1 µA. As can be seen from the figure, the forming voltage for all five devices is between 3.6 V to 3.8 V. The pristine devices have an impedance of > 100 GΩ. The resistance drops after the forming process.

Figure 88 shows the I-V characteristics of five different memristor devices after they have been formed. These devices were set and reset repeatedly using a voltage of 0.8 V and -1.5 V respectively. These curves show that these memristor devices can be set and reset multiple times

without having a lot of variation between switching cycles. This demonstrates the repeatability of memristor write operations.

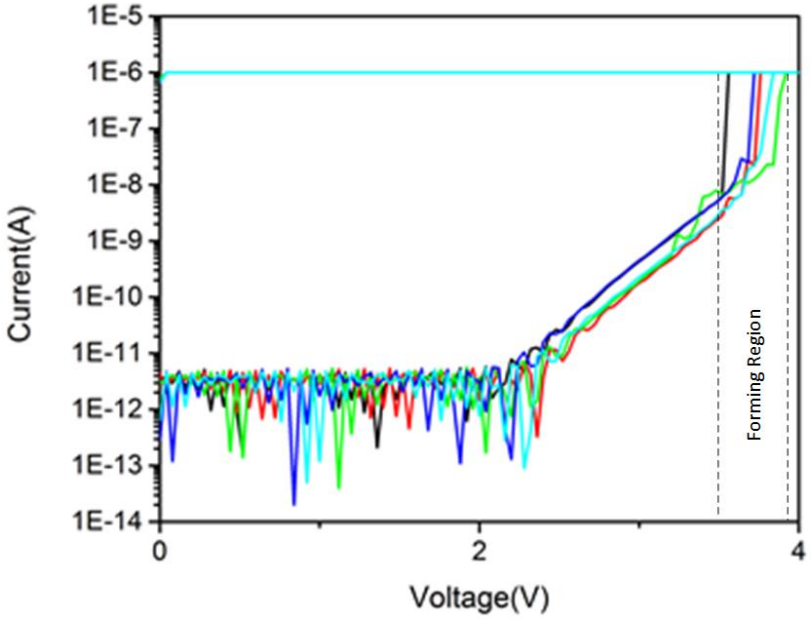


Figure 87: I-V characteristics of five microscale memristor devices

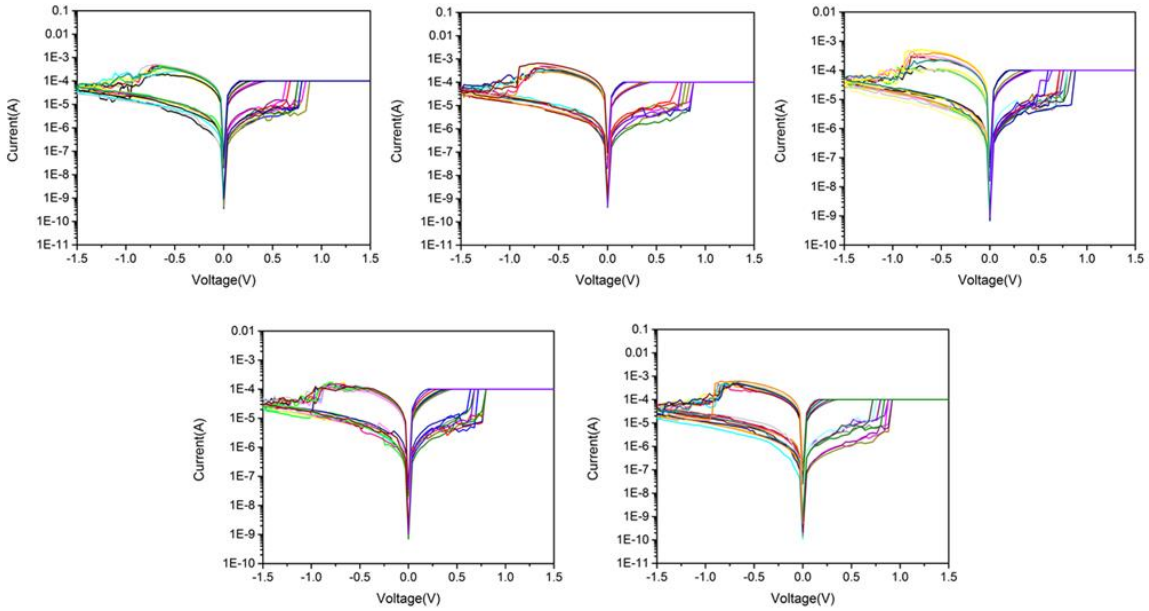


Figure 88: I-V characteristics of five microscale memristor devices obtained from repeated sweeping

4.5.8. Multiple Bits Per Cell

The test procedure in section 4.5.4.5 was used to demonstrate that a single memristor device can achieve multiple conductance levels. The operating range of resistances for the memristors is set to be between 1 KΩ and 10 KΩ. This was dictated by the quantum conductance states of the memristor devices. The measurements performed here demonstrate that the microscale memristor device can achieve thirty-two (5-bits) different resistance levels between 1 KΩ and 10 KΩ and these resistances levels are linear. The linearity of these resistance states is important to perform VMM operations.

Figure 89 shows the measured data for multiple resistance levels of a single memristor device. Here the memristor is programmed to each one of the 32 different levels and each level is read out 1000 times using a readout pulse sequence of 0.2 V and a pulse width of 10 µs as shown in Figure 89(b). The blue curve in Figure 89(b) is the input voltage and the red curve is the output current for a single readout process as an example. Figure 89(a) shows the 32 different levels that are achieved using the microscale memristor device.

From the 1000 readouts of each conductance level, statistical data is collected. Table 8 shows the 32 conductance levels and their corresponding standard deviations obtained from reading these conductance values 1000 times. The fourth column gives the standard deviation values as a percentage of their corresponding target conductance values. Figure 90 shows a plot of these 32 linear conductance levels and the corresponding standard deviations expressed as a percentage of the target conductance value. As can be seen from these results, the memristor read variation is the highest for the lowest conductance state and it reduces as the conductance value increases. However, the worst-case read variation for the microscale memristor device is less than 1% of the target conductance value.

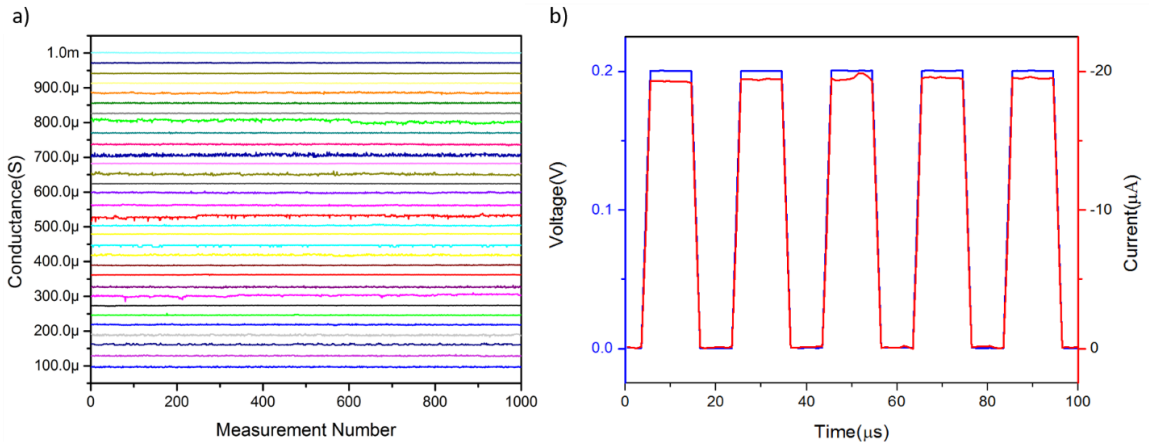


Figure 89: (a) 1000 readings of the 32 different conductance levels, and (b) the reading pulse is a sequence of 0.2 V pulses with a pulse width is 10 µs

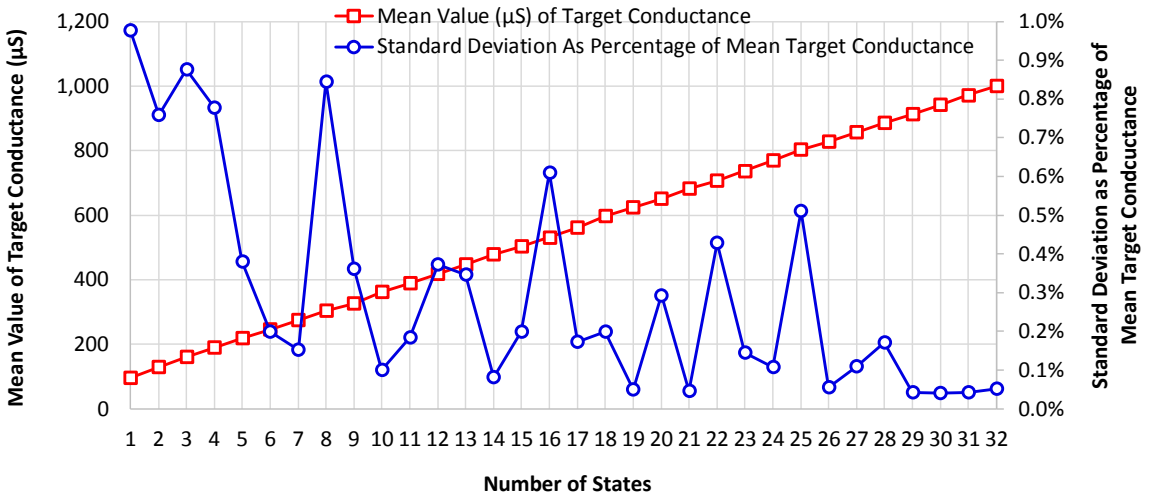


Figure 90: 32 linear conductance levels and their corresponding standard deviations obtained from 1000 readouts expressed as a percentage of the target conductance value

Table 8: 32 conductance levels, their corresponding standard deviations and the standard deviation expressed as a percentage of the target conductance value

Conductance Levels	Mean of Target Conductance value (µS)	Standard deviation (nS)	Percentage Variation as Percentage of Mean Target Conductance
1	96.9	947.0	0.98%
2	129.0	978.0	0.76%
3	161.0	1,410.0	0.88%
4	189.0	1,470.0	0.78%

5	219.0	831.0	0.38%
6	246.0	489.0	0.20%
7	274.0	416.0	0.15%
8	303.0	2,560.0	0.84%
9	327.0	1,180.0	0.36%
10	362.0	365.0	0.1%
11	389.0	716.0	0.18%
12	419.0	1,560.0	0.37%
13	447.0	1,550.0	0.35%
14	479.0	393.0	0.08%
15	504.0	1,000.0	0.2%
16	531.0	3,240.0	0.61%
17	562.0	972.0	0.17%
18	598.0	1,190.0	0.2%
19	624.0	315.0	0.05%
20	651.0	1,910.0	0.29%
21	682.0	320.0	0.05%
22	706.0	3,030.0	0.43%
23	737.0	1,070.0	0.15%
24	770.0	826.0	0.11%
25	804.0	4,110.0	0.51%
26	827.0	463.0	0.06%
27	856.0	940.0	0.11%
28	885.0	1,510.0	0.17%
29	913.0	391.0	0.04%
30	942.0	391.0	0.04%
31	972.0	409.0	0.04%
32	1,001.0	524.0	0.05%

4.5.9. Device-to-Device Variation

The experiment in the previous section demonstrated that a single memristor device can achieve 32 different linear conductance levels. The experiment also demonstrated that the conductance read variation of the memristors are below 1%. The objective in this section is to analyze the memristor device-to-device variation. Five memristor devices were selected and each

of them were set to the same conductance state. Each device was then read repeatedly 1000 times. The results are then plotted and the statistical data is analyzed.

Figure 91 shows a plot of the average value of the five conductance states over 1000 read operations for all five memristor devices. The curves overlap completely which demonstrates that the memristor device to device variation is very minimal.

Figure 92 shows the standard deviations for each conductance state for all five devices. As can be seen from the figure the standard deviation values are very low.

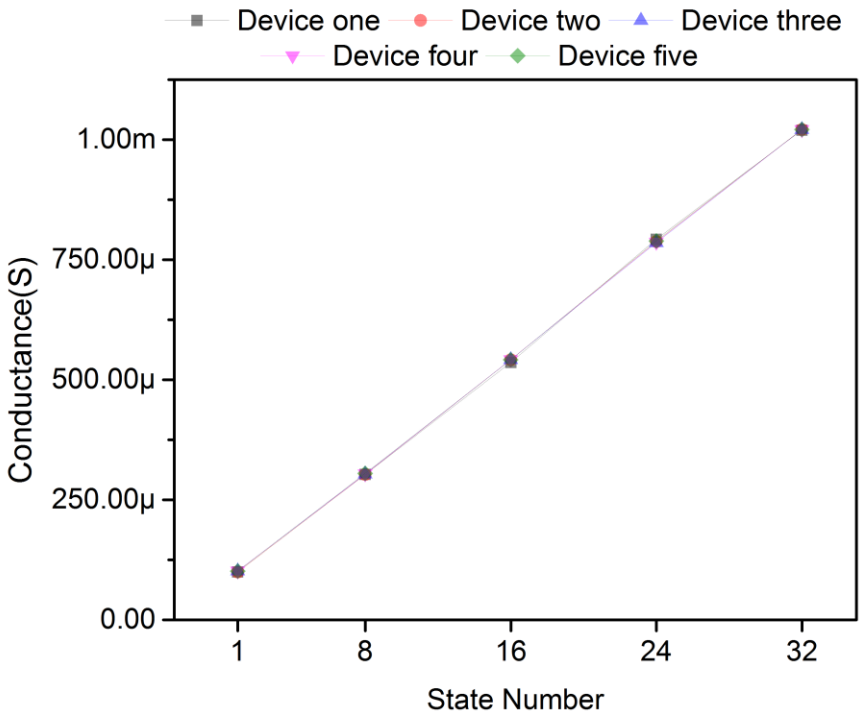


Figure 91: Average conductance values for 1000 read operations for five different devices

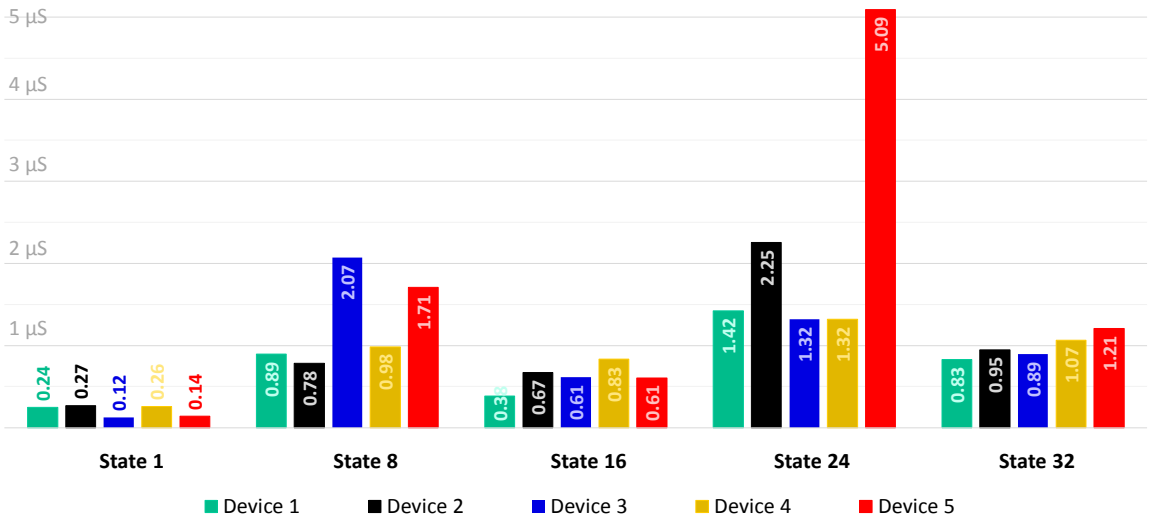


Figure 92: Standard deviations (µS) of 1000 measurements for five conductance states and five devices

The worst case standard deviation that is observed is 5 µS for the 24th conductance state whose mean value is 7.7e-4 S. The standard deviation corresponds to about 0.64% of the target conductance value. The difference between the linear conductance values (step size) for the 32 conductance levels is about 29 µS. These memristor device-to-device conductance variations are lower than the difference between the conductance states and this ensures that there are no overlaps or ambiguities in the conductance values during readouts. This demonstrates that the memristor device-to-device variation is very minimal.

4.5.10. Read Speed Measurements

The test procedure for performing read speed measurements is given in section 4.5.4.4. A function generator is used to input a read voltage pulse to measure the conductance state of the device. The function generator had a rise/fall time of 5 ns. The pulse width was set to 10 ns which corresponds to 100 MHz readout speeds. Figure 93(a) shows the 10ns input voltage applied to the memristor device through the function generator. Figure 93 (b) shows the response of the memristor device in series with the 4.99 KΩ resistor in blue. The red curve shows the voltage

across the 4.99 KΩ resistor. From these voltage readings, the resistor of the memristor is calculated to be 3.09 KΩ.

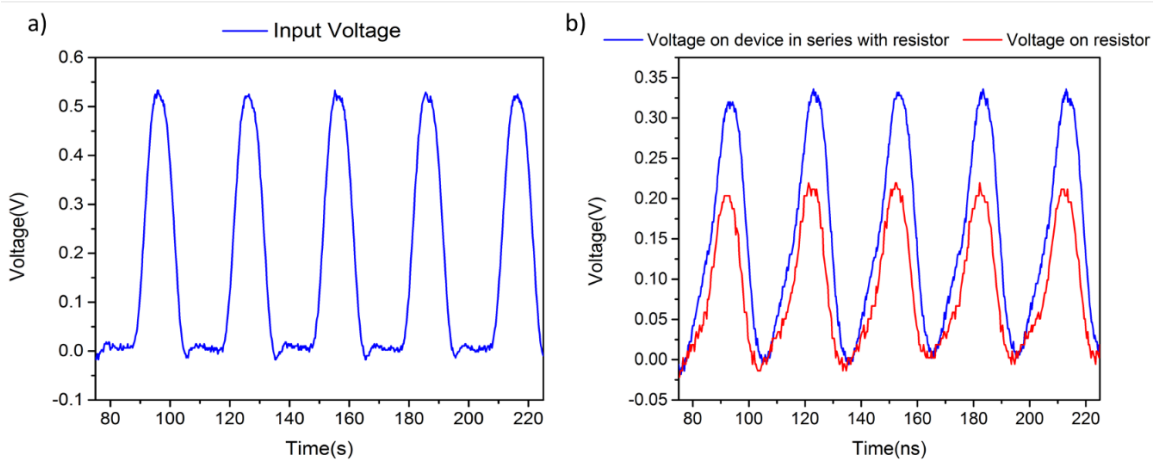


Figure 93: (a) 10 ns input voltage pulse, and (b) voltage drop on device together with a 4.99 KΩ resistor (blue) and the voltage drop on the 4.99 KΩ resistor (red)

4.6. Conclusion

In this part of the dissertation we developed a novel Memristor-CMOS analog co-processor architecture that can perform floating point computation. This is achieved by a bit-slicing the data and using multiple crossbar arrays to perform higher precision computation. We compared the performance of the analog co-processor architecture to existing and future analog and digital processors and it was demonstrated that the Memristor-CMOS analog co-processor can outperform these processors. Microscale memristor devices were fabricated and a characterization board to test these devices were also developed. Using the characterization boards a parasitic model of the microscale memristor device was developed. This model can now be used for developing the analog co-processor models. Other measurements were also performed on the microscale memristor device that showed that the memristor could achieve 5-bits per cell and have a read conductance variation of less than 1% for the worst case and a device-to-device variation of less than 0.64%.

In the next chapter we develop a model for the Memristor-CMOS analog co-processor and we use the model to demonstrate a variety of application simulations.

CHAPTER 5

MEMRISTOR-CMOS ANALOG CO-PROCESSOR MODELING IN PSPICE SYSTEMS OPTION ENVIRONMENT

A Spice model of the memristor-CMOS crossbar array was developed to study the electrical behavior as explained in section 4.2. However, circuit simulation and analysis tools like PSpice alone do not allow us to test the functionality of the analog co-processor with complex inputs. In order to perform these complex simulations, MATLAB/Simulink environments are generally used. MATLAB software is a numerical computing environment specializing in matrix manipulations, while Simulink is a graphical block diagramming environment and has a customizable set of block libraries. The problem with MATLAB/Simulink based simulations is that it is considered to be too ideal even with non-idealities included in them. To circumvent this problem, Cadence and Mathworks have jointly developed a new EDA co-simulation tool called PSpice Systems Option (PSO) [17]. The new tool performs an integrated co-simulation of MATLAB/Simulink and PSpice. It allows for mixed analog and digital simulation, and for computations within Simulink to be mapped into PSpice. This brings credibility to the design and enables a more realistic evaluation of the accuracy, power consumption and acceleration benefits provided by the co-processor.

4.7. PSpice Systems Option Environment Setup

The co-simulation environment being setup uses a combination of MATLAB/Simulink and PSpice tools to perform simulations. The static memristor SPICE model mentioned in section 4.2 is used as the basis for building memristor crossbar arrays in PSpice. The memristors of the

crossbar array are written into by explicitly forcing each memristor to a conductance state by changing the value of the state variable *y*, which is used as a local parameter in the device model.

The next step in the co-simulation environment setup is importing the PSpice memristor-CMOS crossbar array model into the Simulink block diagramming environment. Simulink has a new block called PSpice Block shown in Figure 94, which has been developed for the co-simulation. The Simulink PSpice block has various co-simulation settings as shown in Figure 95. Since different applications use memristor crossbar arrays of different sizes, we can load any PSpice design using the Project File option. The co-simulation setting also has the option of selecting different simulation profiles according to the requirement. The inputs to the memristor crossbar array are read voltages along each row of the crossbar to perform VMM operation. The outputs are currents along each column. These can be selected as shown in the co-simulation setting window in Figure 95. The state variable *y* created as local parameters in the SPICE memristor model has to be declared and used as global parameter to be accessed in the Simulink block diagramming environment. Once these parameters are declared as global variables, they appear in the co-simulation setting window in Simulink as shown in Figure 96. These parameters can be changed from this window which in turn would change the conductance values of the memristors.

Figure 94: PSpice Block in Simulink

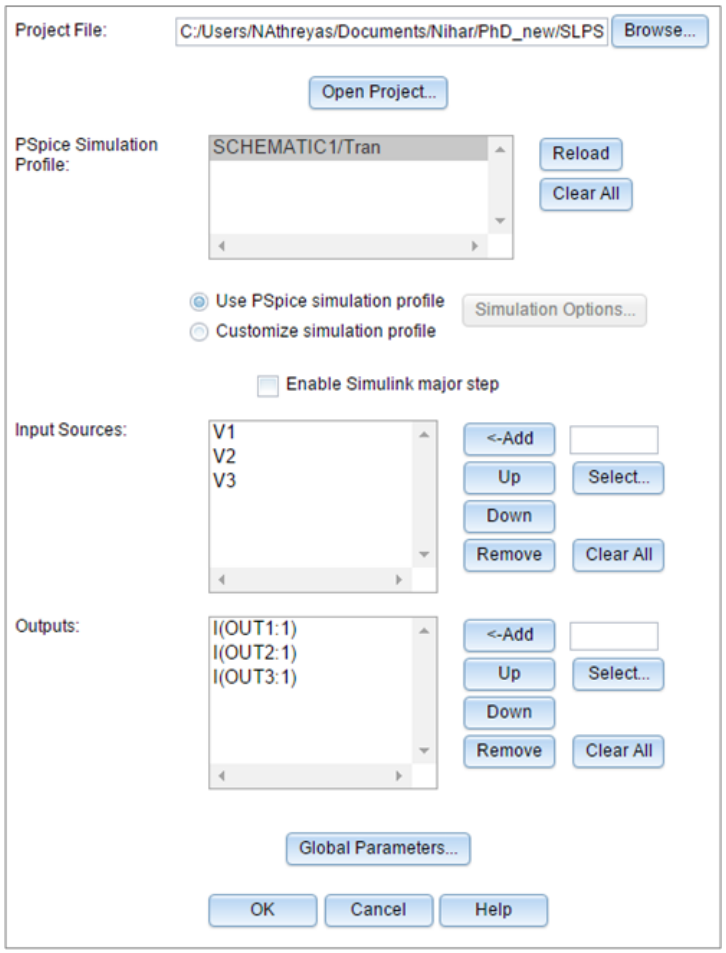


Figure 95: Co-Simulation settings within the Simulink PSpice Block

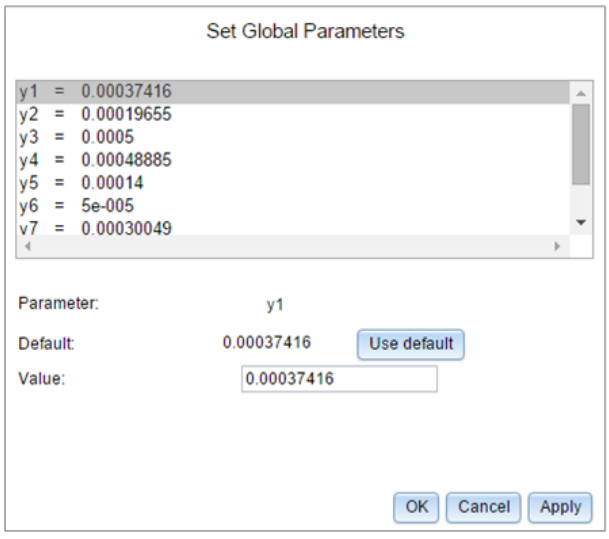


Figure 96: Global Parameters setting window

4.8. Co-Simulation of the Analog Co-Processor

Figure 97 shows the flowchart for the co-simulation of the analog co-processor in the PSpice Systems Option environment. Here, the MATLAB code for an application represents a digital system running an application or algorithm. A computationally intensive part of the algorithm where a VMM operation is needed is first identified. Once this is done, a decision is made as to which kind of operation, a 2D convolution or a general matrix-matrix multiplication needs to be performed by the analog co-processor. As explained in section 4, this decision is made by the application layer. Here, we substitute this by a switch case which decides the mode of operation of the co-processor. The next step is to identify the matrix to be used to perform the VMM operation. Once the matrix is identified, it passes through the blocks of the VMM signal processing chain which are implemented as MATLAB/Simulink blocks. The floating point matrix values are normalized in the normalizing block developed in the MATLAB/Simulink environment as explained in section 4.4.1.2. The normalized matrix values are then mapped to the conductance values of the memristor as explained in section 4.4.1.3. These conductance values are converted to memristor state variable values. Depending on the application and the size of the matrix, a memristor-CMOS crossbar array circuit developed in PSpice is loaded on to the Simulink environment as explained in the previous section. Using the state variables, the matrix is written into the memristor-CMOS crossbar array. The next step is to identify the input vectors. These floating point input vectors are normalized and these normalized vector values are mapped to read voltages by another mapping block developed in MATLAB/Simulink. The read input voltages are supplied along the rows of the crossbar array to perform VMM operation. The VMM operation is performed on the memristor-CMOS crossbar array in PSpice at a circuit level. The outputs of

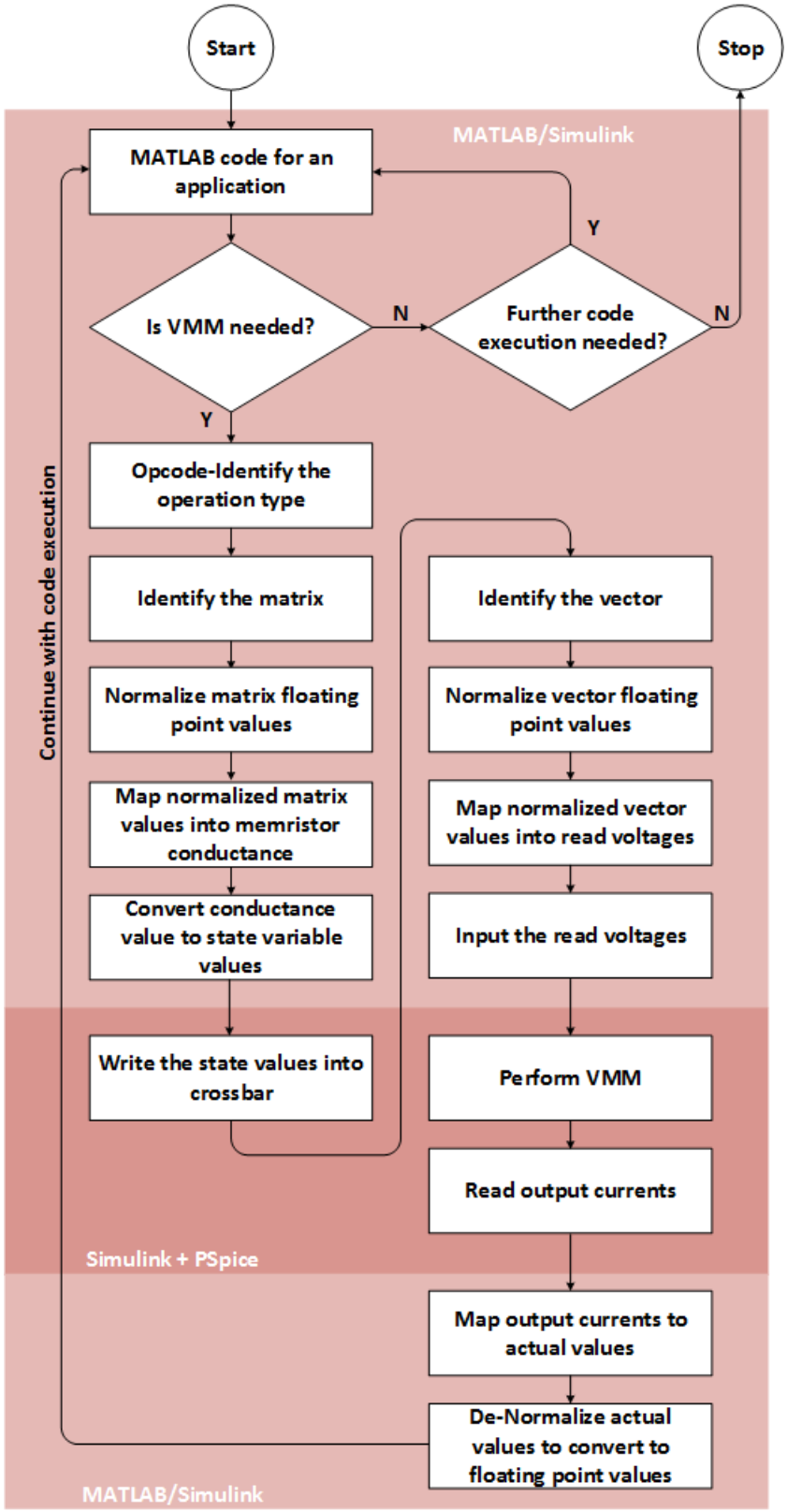


Figure 97: Flowchart for the analog co-processor co-simulation

the VMM operation will be currents on each column which are sampled and brought back to the MATLAB/Simulink environment. These output currents are converted to voltages and are inverse mapped to actual values using the inverse mapping block developed in MATLAB/Simulink. These values are then de-normalized to convert them back to floating point values and the code execution continues until it comes to an end.

4.9. Application Simulation of the Analog Co-Processor Using PSpice Systems Option

The PSpice Systems Option tool is used to perform complex application simulations on the analog co-processor model. The primary objective of the analog co-processor is to accelerate VMM in the context of two major types of operations namely 2D convolution/correlation and general matrix-matrix multiplication. The analog co-processor is being designed for emerging high-performance computing applications like image processing, machine vision, deep learning and scientific simulation. To demonstrate the functionality of the analog co- processor we choose two applications. The first application that we simulate on the analog co-processor is image processing, where we perform different kinds of image filtering which uses the 2D convolution type of operations. We also demonstrate image color transformation and image compression which uses the matrix-matrix multiplication type of operations. The second application simulated on the analog co-processor is the solution to various complex partial differential equations (PDEs) like 2D Navier-Stokes, Poisson and Laplacian equation. The solution to these partial differential equations are achieved through spectral methods or Green's function methods. These methods use the general matrix-matrix multiplication operations.

The analog co-processor model is developed in multiple steps leading to a final model that can perform floating point computation. In the first step of the analog co-processor model, the memristor crossbar array is assumed to be ideal with very high resistance range of $10^5\,\Omega$ to

10^6 Ω. The wire resistance is assumed to be 1Ω. The memristor device is assumed to be analog (i.e. infinite precision). Since the memristor is assumed to be analog, the normalization and de-normalization and the other steps associated with floating point computations are not performed here. These set of assumptions are ideal, and the simulation results obtained in this section will be used as a baseline for comparison of simulation results obtained in the future sections.

Table 9: First Analog Co-Processor model parameters and their values

Analog Co-Processor Model Parameter	Value
Low Resistance State	1e5Ω
High Resistance State	1e6Ω
Number of Bits per Memristor Cell	Analog
Wire Resistance	1Ω

4.9.1. Image Processing Application Simulation

The first application simulation that is performed using the first analog co-processor model is image processing. Three different image processing applications are simulated in the PSpice Systems Option model of the analog co-processor. Figure 98 compares an example simulation flow of implementing a linear image filtering operation in MATLAB and PSpice Systems Option. The top row shows the MATLAB simulation flow. Here an image is loaded onto MATLAB. The image is cropped to consider a region of interest and then converted to grayscale. The MATLAB function is used to perform edge detection on the input image and the output is displayed. The problem with this simulation flow is that the edge detection operation is performed in MATLAB which is considered too ideal even with non-idealities included. To resolve this issue, PSpice Systems Option tool can be used. The simulation flow for this is shown in the bottom row. Here the same initial steps of loading, cropping and conversion to grayscale are followed but instead of performing the VMM operation in MATLAB, the PSpice Systems Option

tool calls the PSpice memristor-CMOS crossbar array circuit model to perform VMM. This gives more credibility to the simulation as it is now being performed at a circuit level. As shown in section 3.5, PSNR and SSIM are used as the performance metrics for these image processing algorithms.

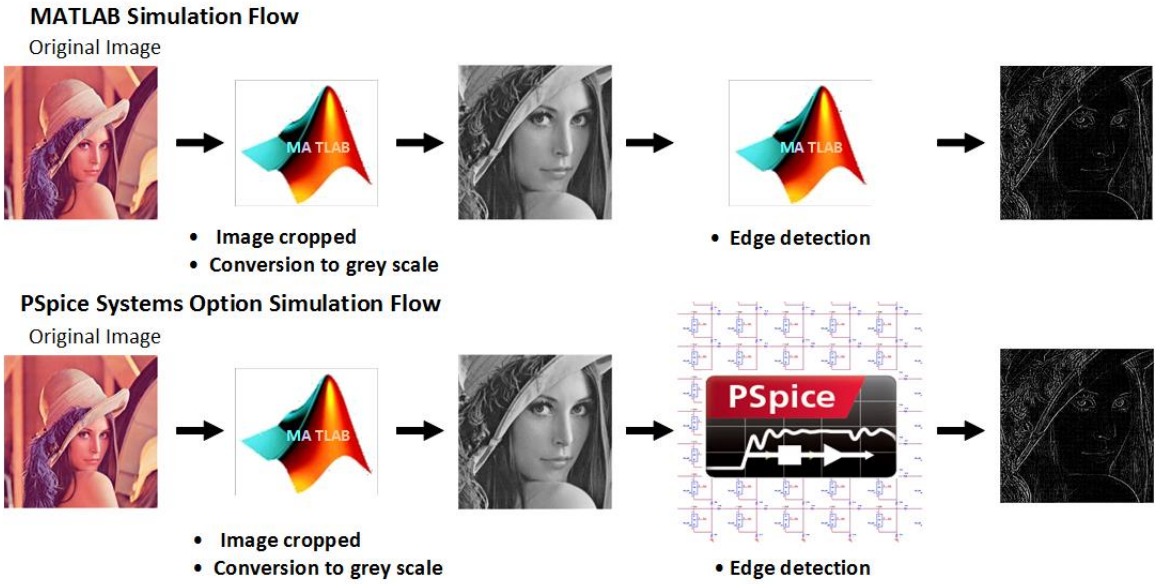


Figure 98: Comparison of the MATLAB simulation flow and PSpice Systems Option simulation flow

5.3.1.1. Image Filtering Simulation Results

In this application simulation, the 2D convolution operation is performed on the analog co-processor model developed in PSpice Systems Option environment by casting it as a VMM operation. Here we have considered seven different image filtering kernels each of size 3 x 3. Each kernel matrix is first converted to a column vector of size 9 x 1. Since we have 7 such kernels we get a 9 x 7 matrix of kernel values. These 9 x 7 kernels values are mapped to memristor conductance values using the mapping block developed in MATLAB/Simulink. These are further converted to the state variable values and are written into a 9 x 7 memristor-CMOS crossbar array.

The input images which must be filtered are converted into vector values and are mapped to memristor-CMOS crossbar array read voltages using the mapping block developed in MATLAB/Simulink. These voltages are input into the memristor-CMOS crossbar array to perform VMM operation in PSpice environment. The output currents are sampled and brought back to the MATLAB/Simulink environment where they are inverse mapped to convert them to actual pixel intensity values to be displayed as output filtered images. The image processing results obtained using the analog co-processor PSpice Systems Option model are compared to MATLAB results which are ideal. The seven image filter kernels that were chosen for the application simulation were a simple averaging filter, a circular averaging filter, Gaussian lowpass filter, a Laplacian filter, Laplacian of Gaussian filter, Prewitt horizontal edge emphasis filter and Sobel horizontal edge emphasis filter [64]. Figure 99 and Figure 100 show the comparison of implementing these seven image filtering operations in MATLAB and PSpice Systems Option analog co-processor model. The top row of each figure shows the MATLAB simulation results and the bottom row shows the PSpice Systems Option results.

As can be seen from the figures, visually and qualitatively both the MATLAB generated outputs and the PSpice Systems Option outputs are very similar. For each filtering operation, we also compare the performance between MATLAB and PSpice Systems Option quantitatively by using two performance metrics, PSNR and SSIM. The PSNR values for the seven filtering operations are 91.62dB, 74.69dB, 71.73dB, 57.18dB, 57.68dB, 75.57dB and 66.76dB respectively. The PSNR value of the output image of a standard image compression algorithm like Joint Photographic Experts Group 2000 (JPEG2000) in a digital system can range from 30 dB to 50 dB [25] and the quality of this compressed image is accepted everywhere. In comparison, the PSNRs for all the seven outputs are above 50 dB and, hence, we can conclude that the performance of

the analog co-processor is good. The SSIM values are all close to 1, which indicates that the MATLAB outputs and the analog co-processor outputs are structurally similar as well.

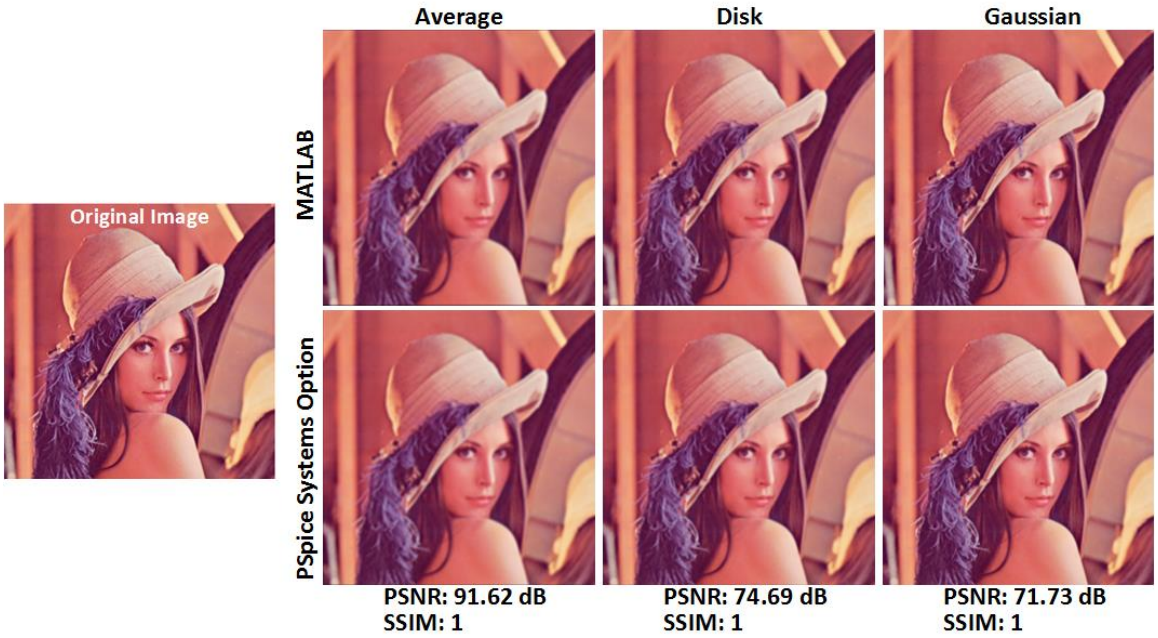


Figure 99: Performance comparison of three different averaging filter operations performed using MATLAB and PSpice Systems Option

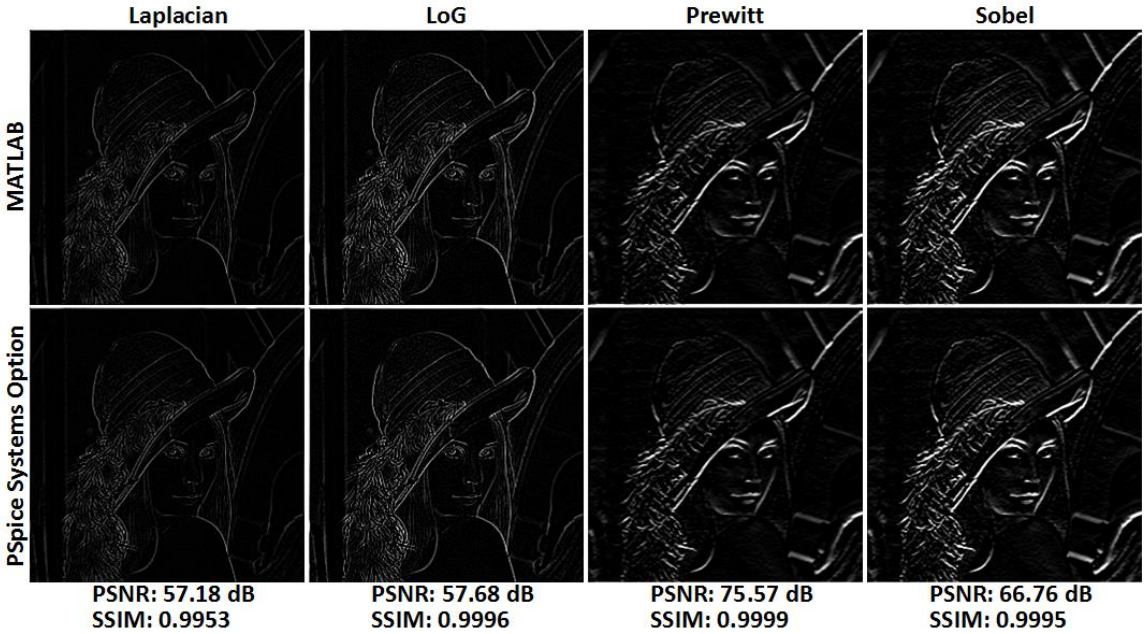


Figure 100: Performance comparison of three different edge detection filter operations performed using MATLAB and PSpice Systems Option

5.3.1.2. RGB to YUV Color Transformation Simulation Results

YUV is a color space that is used as a part of an image pipeline. It encodes a color image by taking human perception into account. This allows for reduced bandwidths for chrominance components thereby enabling transmission errors or compression artifacts to be more efficiently masked than a direct RGB transmission. The color transformation from RGB to YUV can be performed using the equations below. Here *Y* is the luminance component, *U* and *V* are the chrominance components [26].

$$Y = 0.299R + 0.587G + 0.114B \quad (31)$$

$$U = -0.147\,R - 0.289\,G + 0.436\,B \quad (32)$$

$$V = -0.615\,R - 0.515\,G + 0.1\,B \quad (33)$$

In this application simulation, the operation being performed is a general vector-matrix multiplication. The 3 x 3 coefficients from the conversion equations above are mapped to memristor conductance values and then converted into state variable values using the MATLAB/Simulink mapping blocks. These values are then written into a 3 x 3 memristor-CMOS crossbar array. The input images that have to be transformed are converted into vector values and are mapped to memristor-CMOS crossbar array read voltages using the mapping block developed in MATLAB/Simulink. These voltages are input into the memristor-CMOS crossbar array to perform VMM operation in PSpice Systems Option co-simulation environment. The output currents are sampled and brought back to the MATLAB/Simulink environment where they are inverse mapped to convert them to actual pixel intensity values to be displayed as output transformed images.

Figure 101 shows the input RGB image and the *Y*, *U* and *V* components generated from both MATLAB and PSpice Systems Option tool. Qualitatively both the MATLAB and PSpice Systems Option results look similar. The PSNR for the analog Y component is 91.78dB, analog *U* component is 98.27dB and analog *V* component is 88.51dB. The SSIM values for all the components are 1. These PSNRs and SSIMs were measured by taking the MATLAB simulated outputs as true images.

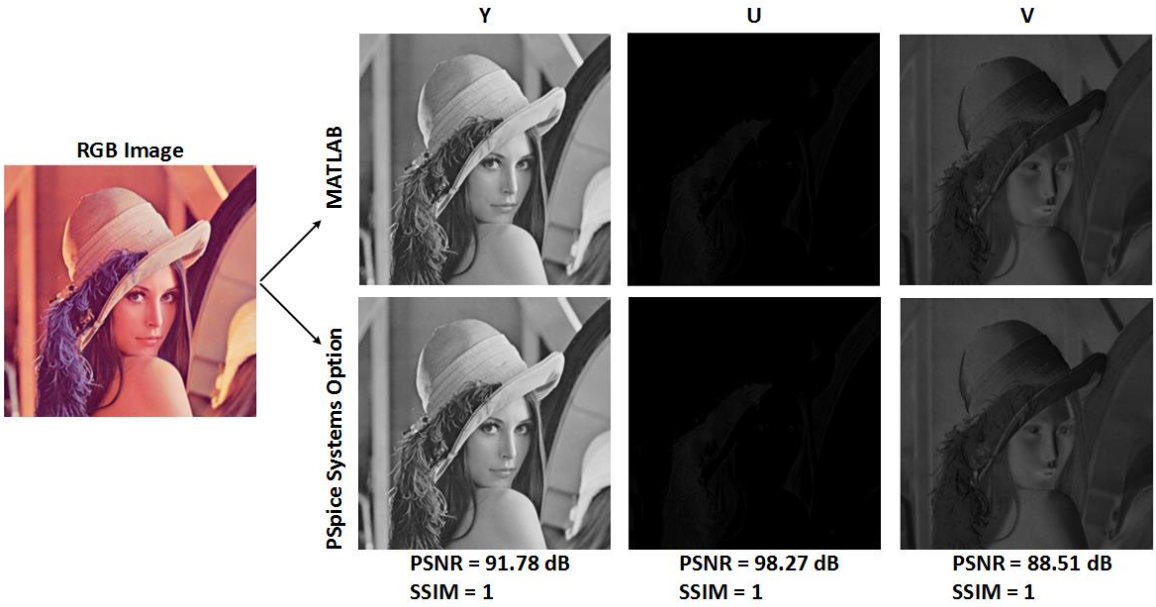


Figure 101: Comparison of RGB to YUV color transformation on the Lena image in MATLAB and PSpice Systems Option

The high values of the PSNRs and SSIMs indicate that the analog co-processor can be used very efficiently for performing color image transformations which are used very frequently in emerging machine vision applications.

5.3.1.3. Discrete Cosine Transform Image Compression

The discrete cosine transform (DCT) is closely related to the discrete Fourier transform. It is a separable linear transformation; that is, the two-dimensional transform is equivalent to a one-dimensional DCT performed along a single dimension followed by a one-dimensional DCT in the other dimension. The definition of the two-dimensional DCT for an input image *A* and output image *B* is

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \qquad (34)$$

$$0 \le p \le M-1 \; and \; 0 \le q \le N-1$$

$$\text{Where, } \alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 0 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 0 \leq q \leq N-1 \end{cases}$$

M and *N* are the row and column size of *A*, respectively. If DCT is applied to real data, the result is also real. The DCT tends to concentrate information, making it useful for image compression applications [51][77].

In this application simulation, an 8 x 8 DCT matrix '*D*' is constructed in MATLAB. The 64 DCT coefficients are mapped into memristor conductance values which are then converted to state variable values. These values are written into an 8 x 8 memristor-CMOS crossbar array. The input image is also divided into 8 x 8 blocks. Each 8 x 8 block is then converted into a 64 x 1 column vector. These pixel intensity values are mapped to read voltages of the memristor-CMOS crossbar array to perform VMM. For each block '*A*,' we calculate the 2D DCT as $D*A*D^T$. The image can then be recovered by taking the inverse transform as D^T*B*D where *B* is the transformed image. The matrix multiplication operations are performed on the memristor-CMOS analog co-processor model in PSpice Systems Option environment. This operation is different from 2D convolution that was performed for the image filtering case. Here we are using VMM to perform two 2D matrix multiplications. Figure 102 shows the color image transform being performed. The input Lena image is operated on by 8 x 8 DCT coefficients to produce a transformed image. The transformed image is again operated on by the DCT coefficients to perform the inverse transform which recovers the image as shown in the figure. Both the transform and its inverse involve performing two 2D matrix multiplication operations which are performed on the analog co-processor model. The PSNR and SSIM numbers are obtained by comparing the recovered image to the original input

image. The PSNR obtained is 61.24dB and SSIM is 1. These high values indicate the excellent performance of the analog co-processor for image transform applications.

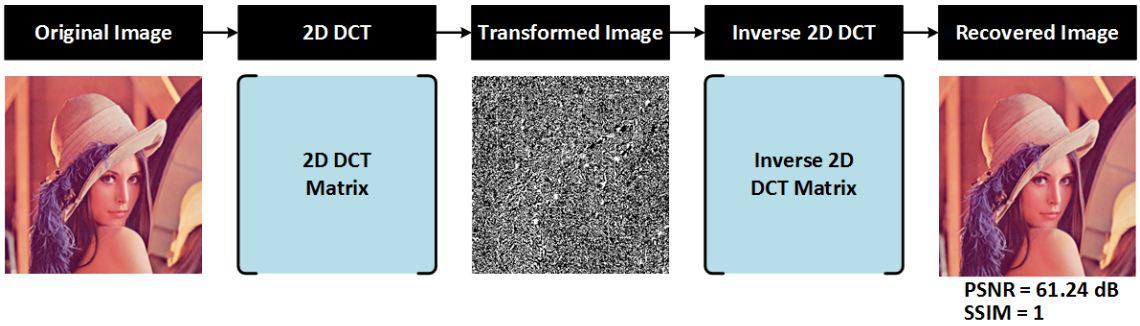


Figure 102: DCT and IDCT of color image

4.9.2. Application Simulation of Solution to Partial Differential Equations

The second application simulation that is considered is the solution to PDEs. Current PDE solution methods are inefficient and often intractable due to limitations associated with discrete variable encoding and serial processing. However, the analog co-processor can be a very effective tool in solving PDEs by invoking VMM. Computational fluid dynamics, particularly as modeled through Navier-Stokes, is perhaps the most persistent class of PDEs used for scientific simulation in academia and industry, with broad applications in industry verticals such as fusion energy, automotive, and systems biology. Additionally, solutions to Laplace and Poisson equations were simulated to highlight the value proposition of analog computation while simplifying the physics involved. Several different methods such as Fourier spectral method and Green's function method were demonstrated to solve these PDEs.

5.3.2.1 2D Navier-Stokes Equation

A Navier-Stokes equation is used to model the flow of viscous fluids. This equation assumes that the stress in the fluid is the sum of a diffusing viscous term which is proportional to the gradient of the velocity and a pressure term. Navier–Stokes equations are useful because they

describe the physics of many phenomena. They can be used to model the weather, study the waterflow in a pipe and airflow around a wing.

Consider the general incompressible Navier-Stokes equations for fluid dynamics:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{35}$$

$\nabla \cdot \mathbf{u} = 0$ where **u** is the flow velocity, p is the pressure and μ is the viscosity.

The numerical method to advance the solution from timestep n to n+1 for a periodic boundary condition is as follows [95]:

i. $\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^{n}_{trace}$ where $\mathbf{u}^{n}_{trace}$ is the velocity at a point traced back along the streamline a distance $\Delta t\mathbf{u}$. This accounts for the advective nonlinear part of the equation $\mathbf{u} \cdot \nabla \mathbf{u}$.

ii.Find $\hat{\tilde{\mathbf{u}}}^{n+1}$ through the DFT of $\tilde{\mathbf{u}}^{n+1}$.

iii. $\hat{\mathbf{u}}^{n+1} = \frac{1}{(1+\Delta t \mu \mathbf{k} \cdot \mathbf{k})} \left[\hat{\tilde{\mathbf{u}}}^{n+1} - \frac{(\hat{\tilde{\mathbf{u}}}^{n+1} \cdot \mathbf{k})}{\mathbf{k} \cdot \mathbf{k}} \mathbf{k} \right]$ accounts for the pressure and viscous term

iv.Find $\mathbf{u}^{n+1}$ through the IDFT of $\hat{\mathbf{u}}^{n+1}$

The above solution is based on the spectral method as the flow velocity is computed by calculating the Fourier transform and its inverse. A MATLAB routine to solve the Navier-Stokes equation based on the above steps is first developed. The computationally intensive parts of the solution are step ii and iv which compute the DFT and IDFT. These steps are offloaded to the analog co-processor. A 16-point DFT is considered for this application simulation. The 16 x 16 DFT coefficient matrix is mapped into the memristor conductance values and then into state variable values by the mapping block developed in MATLAB/Simulink. These values are written into a 16 x 16 memristor-CMOS crossbar array in the PSpice Systems Option co-simulation environment. The horizontal and vertical components of the velocity vectors are first initialized. These vectors are

mapped into read voltages using the mapping block. These read voltages are used to perform VMM which results in the Fourier transform of the input velocity vectors. The VMM outputs are brought back to the MATLAB/Simulink environment to be inverse mapped. The pressure and viscous terms are accounted for, as shown in step iii. The IDFT operation is performed using the memristor-CMOS crossbar array in PSpice Systems Option environment. The VMM outputs are brought back to the MATLAB/Simulink environment for inverse mapping. This process is repeated for multiple time steps.

In this application simulation, the Navier-Stokes equation is solved for 150 time steps in both an ideal MATLAB environment and in the PSpice Systems Option co-simulation environment. The solution to each time step is the flow velocity of the particles. An error value between the ideal MATLAB outputs and the more realistic PSpice Systems Option outputs are calculated for each time step.

Figure 103 shows the comparison of the solution to 2D Navier-Stokes equation being implemented in MATLAB and PSpice systems option for one timestep. The arrows represent the particles of a fluid. The solution to Navier-Stokes equation is the flow velocity. The magnitude of the velocity is represented by the size of the arrow and the direction is represented by the direction of the arrow. The third window shows the error in the horizontal and vertical components of the velocity vector between the MATLAB result and the PSpice Systems Option result. For the time step shown in Figure 103, the error in the horizontal velocity component is 0.51% and in the vertical component is 0.38%. The average step error over 150 time steps for the horizontal component was 1.58% and for the vertical component was 1.19%. This application simulation demonstrates that the analog co-processor can be used to solve complex PDEs like Navier-Stokes with very small error.

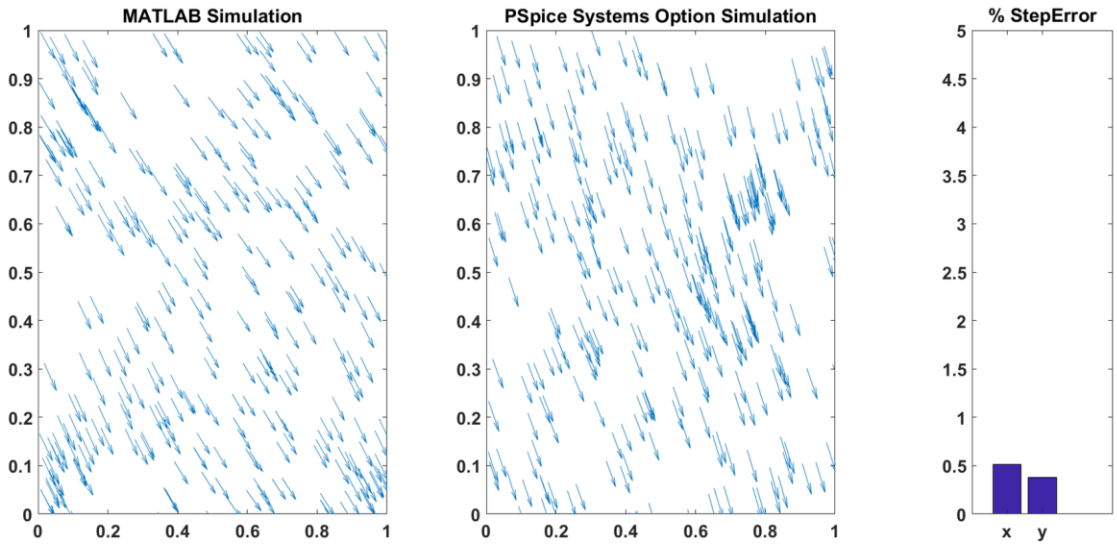


Figure 103: Comparison of solution to 2D Navier-Stokes equation

5.3.2.2 2D Poisson Equation

The second PDE that is simulated is the 2D Poisson equation. This PDE is solved via its approximate discrete Green's function [23]. The Green's function for a partial differential equation can be considered as the impulse response of that partial differential equation. The Green's function is not normally a viable way to solve a PDE on a traditional computer. But new hardware, such as memristors, opens new avenues for PDE solution. In this target application case, we consider the solution to the Poisson equation which is a partial differential equation of the elliptic type with a broad utility in mechanical engineering and theoretical physics. It can be used to describe the potential field caused by a charge or mass density distribution. Once the potential field is known, we can calculate the associated gravitational or electrostatic field.

The Poisson equation is given as, $\nabla^2 T = S$. Any general PDE has a formal mathematical solution that involves the Greens function, $T(\mathbf{x}) = \int_{\Omega} \mathbf{G}(\mathbf{x},\mathbf{x}')S(\mathbf{x}')d\mathbf{x}'$ where $\mathbf{G}(\mathbf{x},\mathbf{x}')$ is the Green's function. Every PDE has a particular Green's function. The 2D Poisson equation Green's function is $\mathbf{G}(\mathbf{x},\mathbf{x}') = \frac{1}{2\pi}\ln(|\mathbf{x}-\mathbf{x}'|)$. If the integral over the domain is approximated on a discrete mesh

with *N* mesh cells/elements, then calculating the solution $T(\mathbf{x})$ in one cell requires summing over all the N-1 other cells and solving for all the solution unknowns (in all the cells) which requires order N^2 operations. This is too expensive because it is known that it is possible to find the solution to this problem in O(*N*) time. For this reason, the Green's function is rarely used as a primary solution technique for solving PDEs. The cost can also be understood in terms of linear algebra and matrices. Once discretized, a PDE produces a matrix problem $A\mathbf{x} = b$, where A is an N x N sparse matrix with only O(*N*) non-zero entries, *b* is a source vector of *N* known values, and *x* is the unknown solution vector (with *N* entries). The Green's function, $\mathbf{G}(\mathbf{x}, \mathbf{x}')$ is the equivalent of the matrix inverse. It is a full matrix and requires N^2 operations to multiply by the source and get the solution, via $\mathbf{x} = A^{-1}b$.

The analog co-processor makes the Green's function approach viable again, because it performs the matrix multiplication operations in the analog domain. This means that all the N^2 VMM operations can be performed in a single cycle on a memristor-CMOS crossbar array. For this application simulation, we considered the solution of a 2D Poisson PDE where three point-source terms were present. This is equivalent to finding the electric potential that results from three point charges distributed on the plane. It is also equivalent to finding the temperature in a large surface that has three point heaters/coolers. The resolution of the problem considered was 128 x 128. The size of the mesh was fixed at 16 x 16. The total number of meshes would be 64. The Green's function matrix of size 16 x 16 is formulated and the values of this matrix are mapped into memristor conductance values which are further converted into state variable values.

These values are written into the memristor-CMOS crossbar array in the PSpice Systems Option environment. The input vector values are converted into read voltages and are input into the memristor-CMOS crossbar array to perform VMM. The VMM outputs are brought back to the

MATLAB/Simulink environment to be inverse mapped. This represents the Green's function method of solving the PDEs. For comparison, the 2D Poisson PDE is solved using the same Green's function approach in MATLAB. Figure 104 shows the solution to the PDE obtained using the PSpice System Option model of the analog co-processor. The scale on the right indicates the intensity of the electrostatic field due to the three point charges. Figure 105 shows the residual plot which indicates the number of iterations it takes the analog co-processor to converge to a solution with an error tolerance of 10^{-10}. For a problem size of 128 x 128 and a Green's function mesh size of 16 x 16 the analog co-processor takes 355 iterations to converge with an average step error of 4.85% when compared to an ideal MATLAB simulation. The MATLAB simulation of the same PDE takes 329 iterations to converge to a solution. It was shown in section 4.4.5 that the theoretical performance of the analog processor is 15x faster than the projected 5nm GPU for a 32-bit operation. Assuming this performance, the analog co-processor converges to the solution of a Poisson PDE 13.9 times faster than a projected 5nm GPU.

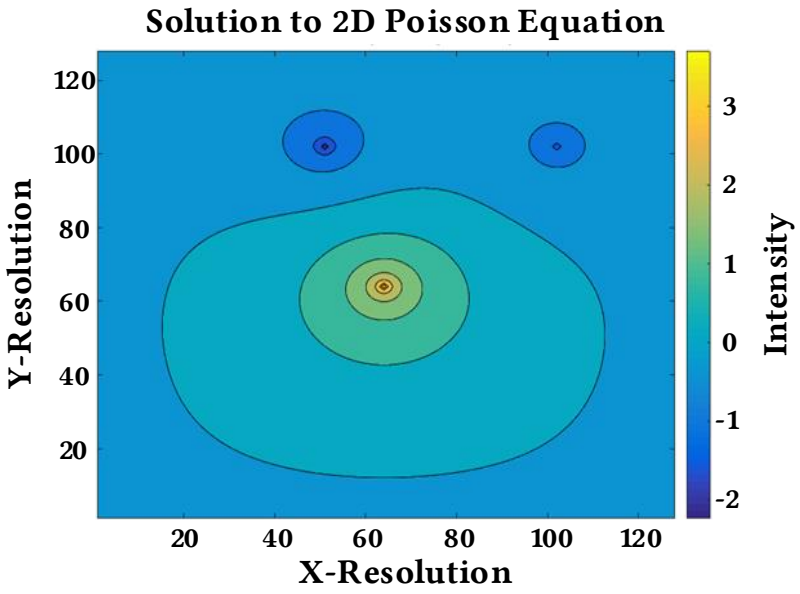


Figure 104: Solution to 2D Poisson PDE with a resolution of 128 x 128

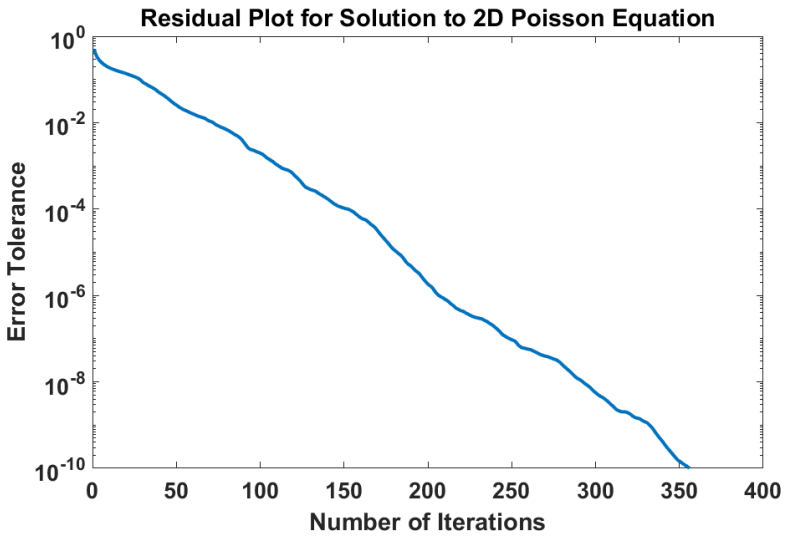


Figure 105: Residual plot for the solution to 2D Poisson PDE with a resolution of 128 x 128

5.3.2.3 2D Laplace Equation

The third PDE solved in the application simulation is the Laplace's equation, $\nabla^2 T = 0$ on a 2D unit square, with boundary values set to zero on 3 sides of the square and 1 on the top of the square. There are no source terms in this equation, so the boundary conditions entirely set the solution. This problem is equivalent to finding the temperature in a plate where the edge temperatures are controlled or the shape of a soap bubble surface when the edge of the bubble is controlled (say by a wire rim). In this problem, the Green's function has a different function expression for every solution unknown. Computing the Greens function exactly for this problem is very expensive, and takes O(N^3) operations, and it cannot be sped up by using an analog VMM. We will therefore use an approximate Green's function. In this application simulation, we use the same Green's function that was used for a Poisson PDE. The free-space 2D Green's function that ignores boundary conditions altogether. Since the Green's function is not exact, it is used as a

preconditioner to speed up an iterative method. The solution method demonstrated here is Conjugate Gradients (CG) [42]. The iterative algorithm is:

$$r = b - Ax$$
$$z = \tilde{A}^{-1} r$$
$$p = z$$
start loop
$$w = Ap$$
$$\alpha = \frac{p \cdot z}{p \cdot w}$$
$$x = x + \alpha p$$
$$r = r - \alpha w$$
$$z = \tilde{A}^{-1} r$$
$$\beta = \frac{(r \cdot z)_{old}}{r \cdot z}$$
$$p = z + \beta p$$
end loop

The most expensive operation in this iterative procedure is the sparse matrix multiplication, $w = Ap$ and the approximate inverse multiplication $z = \tilde{A}^{-1} r$. The sparse matrix multiplication takes O(*N*) computations to complete each iteration. The approximate inverse multiplication uses the analog co-processor. Note that, even though the algorithm mostly uses conventional digital hardware, the one line using the analog co-processor can have a profound impact on the algorithm. If the approximate inverse is a good approximation, the method takes far fewer iterations to converge. An exact inverse in that location, terminates the CG algorithm in a single iteration. On average, the CG algorithm requires order $N^{1/2}$ number of iterations to solve the Laplace equation problem using a traditional approximate inverse (such as the diagonal of the matrix *A*). This means that, as problem sizes get larger the algorithm takes longer to converge. The total time to solve the Laplace problem with CG scales as O($N^{3/2}$) when the computations per iteration is multiplied by the number of iterations required.

For this application simulation, we considered a Laplace PDE with a resolution of 128 x 128 and a Green's function mesh size of 16 x 16. The total number of meshes were 64. The inverse

of the approximate Green's function matrix $\tilde{A}$ is first formulated in MATLAB environment. This matrix is mapped into memristor conductance values which are then converted to state variable values. The state variable values are written into the memristor-CMOS crossbar array in the PSpice Systems Option environment. The input vector is converted to read voltages which are used to perform VMM operation. The VMM outputs are inverse mapped to obtain the result. Figure 106 shows the solution to the 2D Laplace PDE obtained using the analog co-processor in the PSpice Systems Option tool. As can be seen from the figure, the color bar indicates, as an example, the temperature in a plate when the boundary temperatures are fixed. Near the bottom edge where the temperature is being controlled, the temperature is the highest and as the distance from the bottom edge increases the intensity reduces. Figure 107 shows the residual plot for the solution to the 2D Laplace equation which gives the number of iterations required for the analog co-processor to converge to a solution with an error tolerance of 10^{-10}. The analog co-processor converges to a solution in 344 iterations with an average step error of 4.02% when compared to the ideal MATLAB simulation of the same PDE which converges in 323 iterations. It was shown that the theoretical performance of the analog processor is 15x faster than the projected 5nm GPU performance for a 32-bit operation. Assuming this performance, the analog co-processor converges to the solution of a Laplace PDE 14 times faster than a projected 5nm GPU.

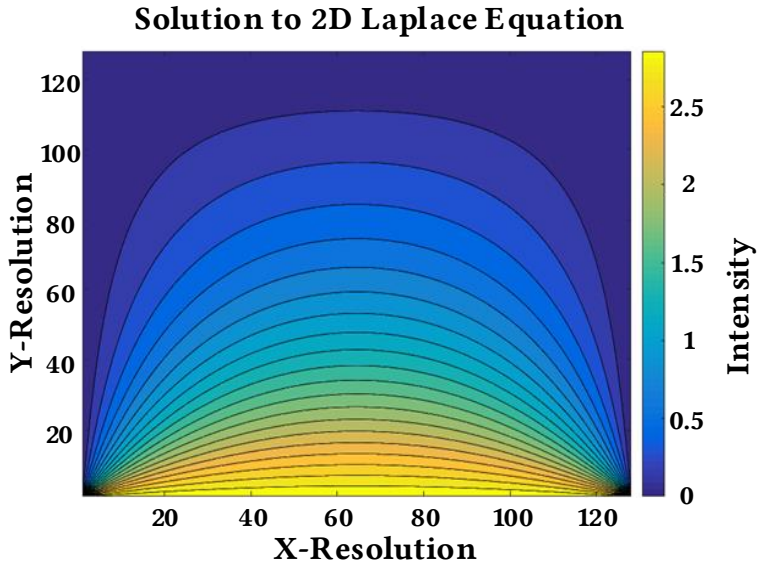


Figure 106: Solution to 2D Laplacian PDE with a resolution of 128 x 128

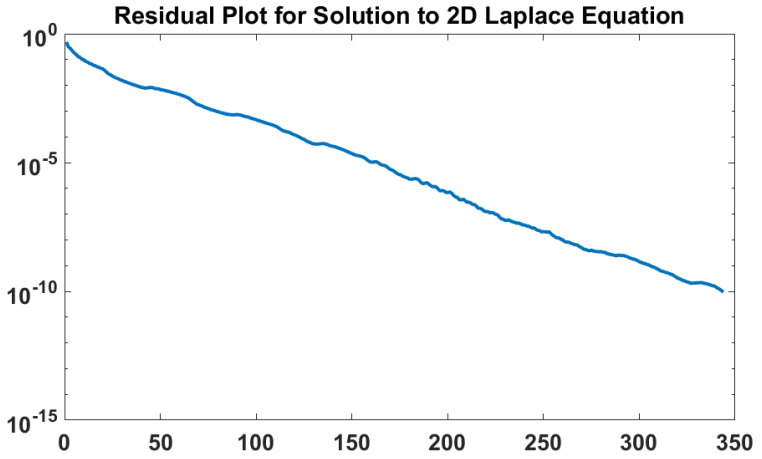


Figure 107: Residual plot for the solution to 2D Laplacian PDE with a resolution of 128 x 128

5.4. Factors Affecting the Design of the Crossbar Array

The SPICE modeling of the memristor-CMOS crossbar array helped identify various challenges in designing the crossbar array of the analog co-processor. The ratio of the memristor LRS to the wire resistance of the crossbar array should be as high as possible to reduce errors at the output. This required ratio increases as the size of the array grows. The memristors tend to be non-linear at the high resistance states but for VMM operation we need to operate the

memristors in their linear region which corresponds to the low resistance states. Operating memristors in the low resistance states leads to lower ratios between memristor LRS and wire resistance. There are practical limitations due to which the wire resistance cannot be decreased beyond a certain point.

It was found that Tantalum oxide memristors can switch continuously from a low-conductance semiconducting to a high-conductance metallic state. At the boundary between these two regimes are quantized conductance states, which indicate the formation of a point contact within the oxide characterized by multistable conductance fluctuations and enlarged electronic noise. A diverse conductance-dependent noise spectrum, including a transition from $1/f^2$ (activated transport) to 1/f (flicker noise) as a function of the frequency f, and a large peak in the noise amplitude at the conductance quantum $G_Q = 2e^2/h$, was observed [111].

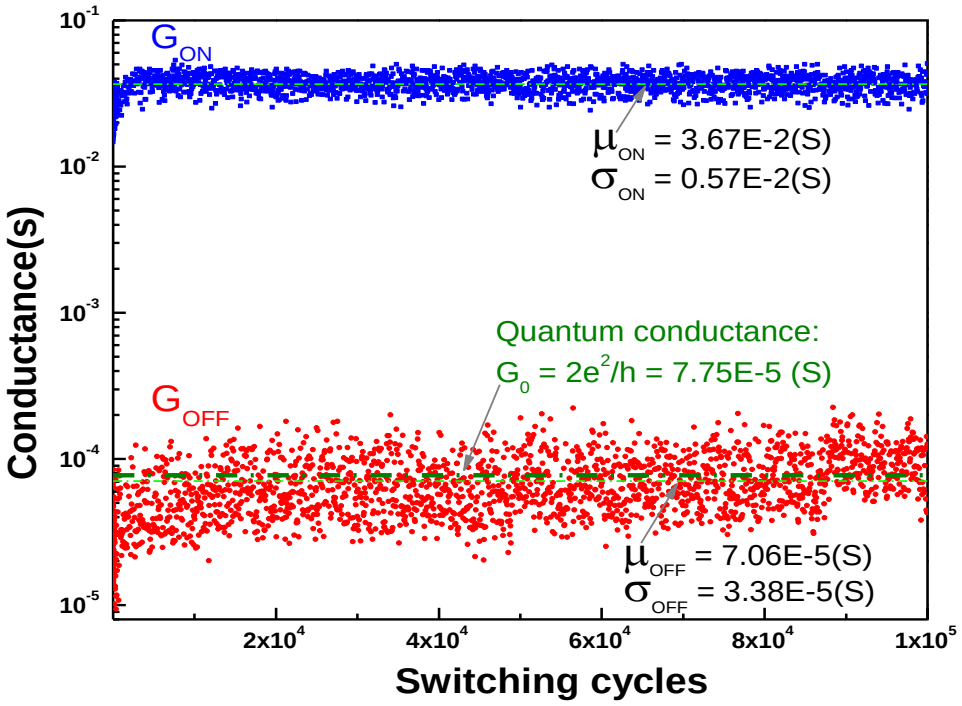


Figure 108: Conductance switching data of a TaOx memristor for 100,000 switching cycles showing larger variance in the regime close to a quantum of conductance [111]

Figure **108** shows the switching data for a TaO_x memristor device. The solid symbols represent conductance data of ON (blue) and OFF (red) states and the green dash dot lines represents the corresponding average values during the 100,000 cycles. Thick olive dash dot line highlights the conductance quantum ($2e^2/h$) which was found to be around 7.75e-5 S for this device. As can be seen from the figure, for the 'ON' state, which is not close to the quantum conductance value, the cycle to cycle variation of the device is very small. The 'OFF' state of the device is very close to the quantum conductance state and hence has a very large variation.

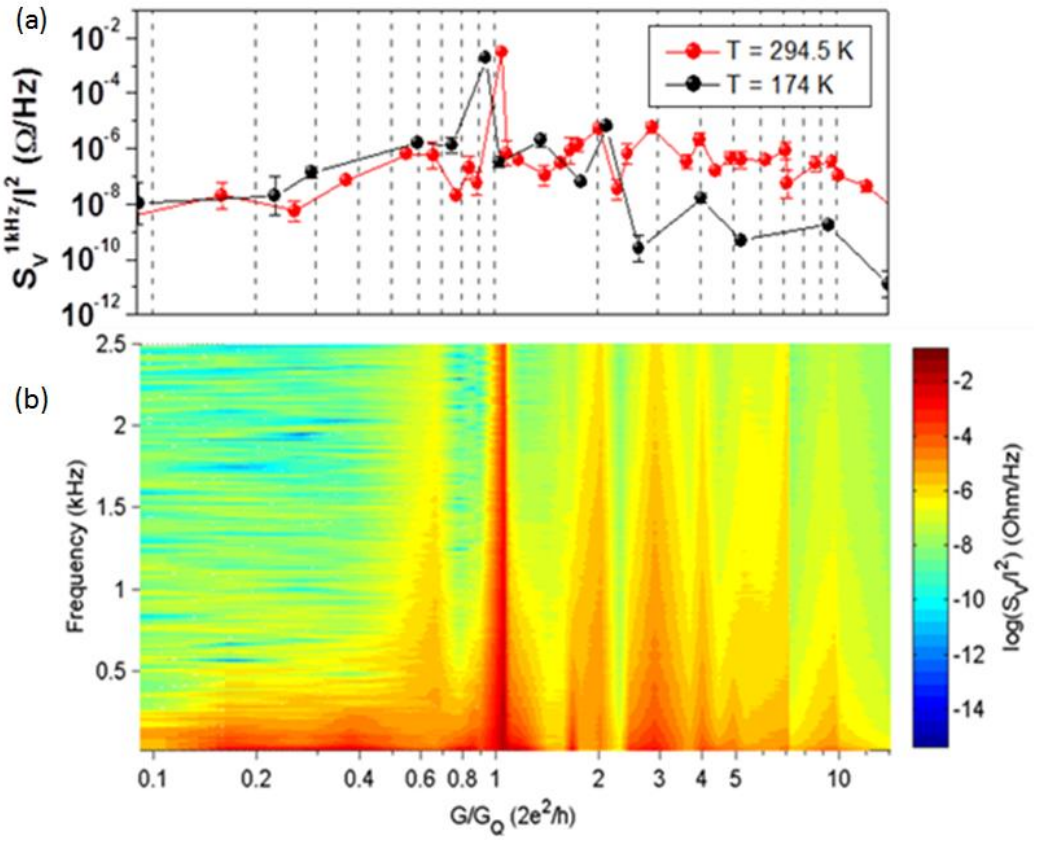


Figure 109: Conductance dependence of normalized noise spectral density [111]

Figure 109 shows the normalized noise spectral density S/I^2 for a TaO_x memristor plotted as a function of G_Q-normalized zero-bias conductance. Figure 109(a) shows S/I^2 at 1kHz for two temperatures, T=295 K and 174K. Figure 109(b) shows a two-dimensional color plot showing S/I^2

as a function of conductance G/G_Q and frequency f at T=295 K. The continuous scale on the color plot was obtained by linear interpolation between measured data points. This representation clearly reveals the markedly increased noise near the first quantum conductance G_Q and the tails of the noise distribution as a function of frequency. In addition, increased noise above the ambient trend can be seen for several other integer multiples of G_Q. The error bars in Figure 109(b) are standard deviations calculated from evaluating S/I^2 at different sourced currents. Small temperature dependence of noise at around $G=G_Q$ and at smaller conductance values (corresponding to a tunneling regime) confirms the present picture of local heating as the main source of noise [111].

From these observations we can conclude that the operating conductance range for memristors for a VMM operation should be more than the quantum conductance value. Hence, in this work we have fixed the conductance range of operation to be between 1e-3 S to 1e-4 S. These values are larger than the quantum conductance value and hence will present a lower variation. These conductance values translate to a resistance range of 1KΩ to 10KΩ. If we consider a wire resistance of 1Ω in the memristor-CMOS crossbar array, then the ratio of the memristor low resistance state to the wire resistance is 10^3. This resistance ratio will prevent us from achieving larger crossbar array sizes and will result in larger errors at the VMM output.

The use of transistors as selector devices in series with every memristor helps mitigate the issue of sneak path currents during the memristor write operation. However, during the VMM operation, which is a memristor read, all the transistors are in the 'ON' state. This creates a complex network of resistors and the voltage drop because of these resistances affects the accuracy of the VMM output. Also, due to the voltage drop along each row of the crossbar array, the memristor cells located at the far end of the array will have lesser read voltage from the driver circuits than the cells that are closer. This will also result in errors at the output.

This combination of low ratios of wire resistance to low resistance state of the memristors and the voltage drops created due to the complex network of resistors will limit the size of the crossbar arrays that can be achieved. In order to address this problem, a practically implementable compensation algorithm that can compensate for the wire resistance, the transistor ON resistance and the voltage drops due to the complex network of resistors that are created during a VMM operation along the rows and columns of the memristor-CMOS crossbar array was developed.

5.4.1. Compensation Algorithms

Various compensation algorithms were considered to improve the performance of the memristor-CMOS crossbar array. We can broadly classify the algorithms that were considered into two categories, post-processing compensation algorithms and pre-processing compensation algorithms. In post-processing compensation, an ideal memristor-CMOS crossbar array is first considered in simulation and a matrix value is written into it. The VMM outputs for this matrix and a set of input vectors are simulated. The same matrix is now written into an actual memristor-CMOS crossbar array and the VMM output is computed for the same set of input vectors. These actual output currents are compared to ideal outputs currents and a regression analysis is carried out to find a relation between the actual and ideal outputs. This is done for each column of the memristor-CMOS crossbar array. Once this relationship is established it is assumed that it holds good for all sets on input vectors and the outputs of the memristor-CMOS crossbar array is compensated in post-processing. However, this compensation technique has major drawbacks especially in terms of practical implementation. In order to compensate for every change in the matrix value, the VMM operation should first be simulated and then compared to an actual output

of the crossbar array. This is very impractical as this defeats the whole purpose of developing an analog hardware accelerator.

Since post-processing compensation techniques don't work well and are not practical we decided to use pre-processing compensation techniques. The goal was to develop a practically implementable compensation algorithm that can compensate for the wire resistance, the transistor ON resistance and the voltage drops due to the complex network of resistors that are created during a VMM operation along the rows and columns of the memristor-CMOS crossbar array.

The compensation algorithm we developed makes use of the information available from the memristor-CMOS crossbar array. The output of each memristor in a crossbar array is a current and these currents sum at each node according to KCL equations to produce a final output current. Measuring the output current at each node gives information about the resistance presented at that node. Figure 110 shows the flowchart of the compensation algorithm that we have developed. Consider a N x N matrix A. This matrix is mapped into the memristor conductance values G_{ideal}. Now consider a 1 x N input vector V composing of all ones. This input vector is mapped into reading voltages V. An input matrix V_{mat} composed of N copies of the input vector is built. Multiplying V_{mat} with G_{ideal} gives ideal node currents. In the analog co-processor design, a separate memristor-CMOS crossbar array will be present which will have current sense amplifiers at each node of the crossbar array built in. The ideal conductance G_{ideal} will be written into this crossbar array. The input vector V composed of all ones will be used to perform a VMM operation on the compensation crossbar array. Using the sense amplifiers on the compensation crossbar array, the actual node currents are measured. Once the node currents are obtained the effective conductance matrix $G_{effective}$ can be computed. The difference between G_{ideal} and $G_{effective}$ which is termed ΔG can be computed. This ΔG value is added to $G_{effective}$ to obtain a new conductance

matrix which we can call $G1_{effective}$ is written into the compensation crossbar array. This process is repeated until the value of ΔG reaches a set tolerance level. This completes the compensation algorithm and this value of the conductance matrix is written into all the other crossbar arrays to perform VMM operation. The use of this compensation algorithm eliminates the dependence of the size of the memristor crossbar array on the ratio of the wire resistance to the memristor low resistance state. This means that, theoretically the size of the memristor crossbar array can now be arbitrary. The compensation algorithm also eliminates all the errors in the VMM output caused due to voltage drops along rows and columns of the crossbar array. We present simulation results in the later sections that demonstrate the effectiveness of the compensation algorithms.

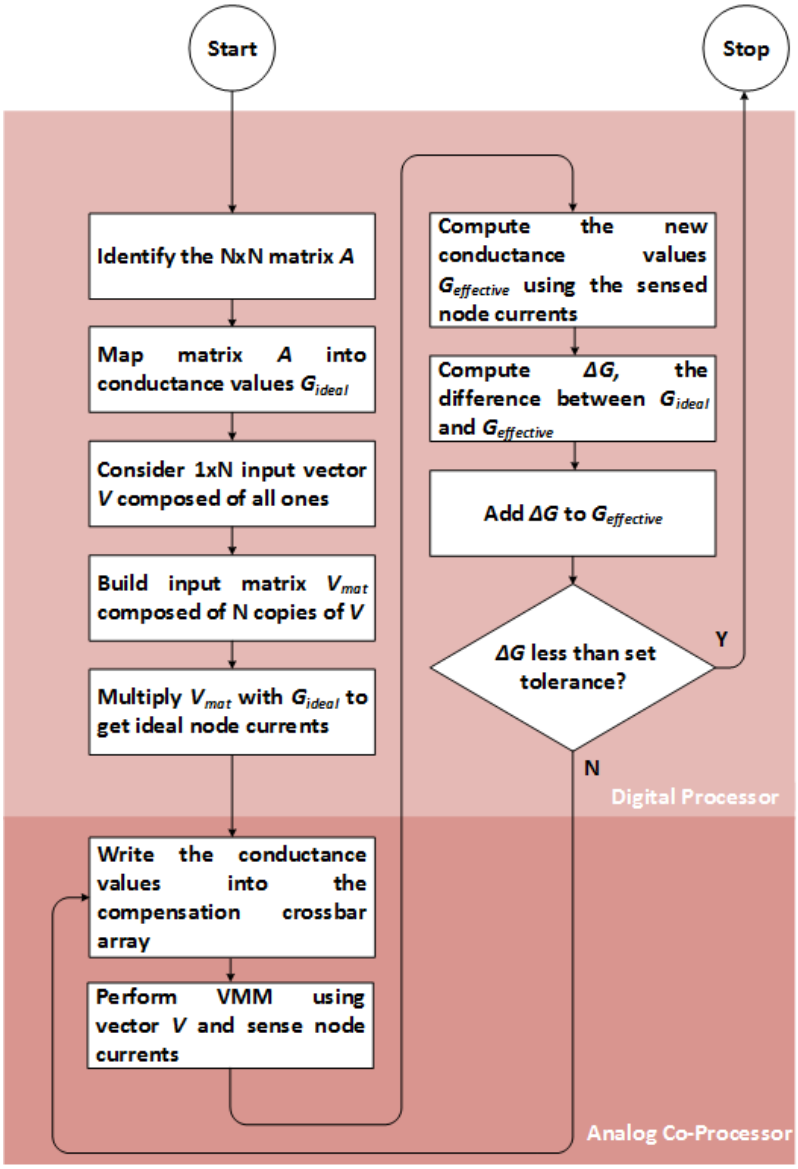


Figure 110: Flowchart describing the compensation algorithm

5.4.2. Implementing the Compensation Algorithm

The compensation algorithm that was developed in the previous section was implemented in the PSpice Systems Option simulation environment. We demonstrate the advantages of using the compensation algorithm through an application simulation of image

filtering on the memristor-CMOS crossbar array using PSO simulation environment. Here we use the memristor crossbar array to perform 2D convolution on images.

As explained in the previous section, due to very high noise and variation at the quantum conductance value of the memristor and its multiples, the operating conductance range of the memristors is reduced to be between 100µS to 1000µS. This translates to resistance values of 1KΩ to 10KΩ. The resistance of the wire of the crossbar array is still assumed to be 1Ω. With these resistance values, the ratio of the memristor resistance state to the wire resistance of the crossbar array in the worst case is 1000 and in the best case will be 10,000.

This is the second step of the analog co-processor model that was developed. This is a more advanced model than the one that was used previously in section 4.9.

Table 10: Second Analog Co-Processor model parameters and their values

Analog Co-Processor Model Parameter	Value
Low Resistance State	1e3Ω
High Resistance State	1e4Ω
Number of Bits per Memristor Cell	Analog
Wire Resistance	1Ω
Compensation Algorithm	Used

Figure 111 shows the comparison of the results between MATLAB and PSO without the use of the compensation algorithm. As can be seen from the figure, the PSNR values for all the filtered images are less than 25dB without the use of the compensation algorithm. The PSNR values obtained here are low. The SSIM values range between -1 and 1 depending on the structural similarity of the two images being compared. As can be seen from the figure, the SSIM values for some of the filtering operations are very low indicating structural dissimilarity. From

these performance metric numbers, it can be concluded that the output of the memristor-CMOS crossbar array is very noisy and has a lot of errors without the use of the compensation algorithm.

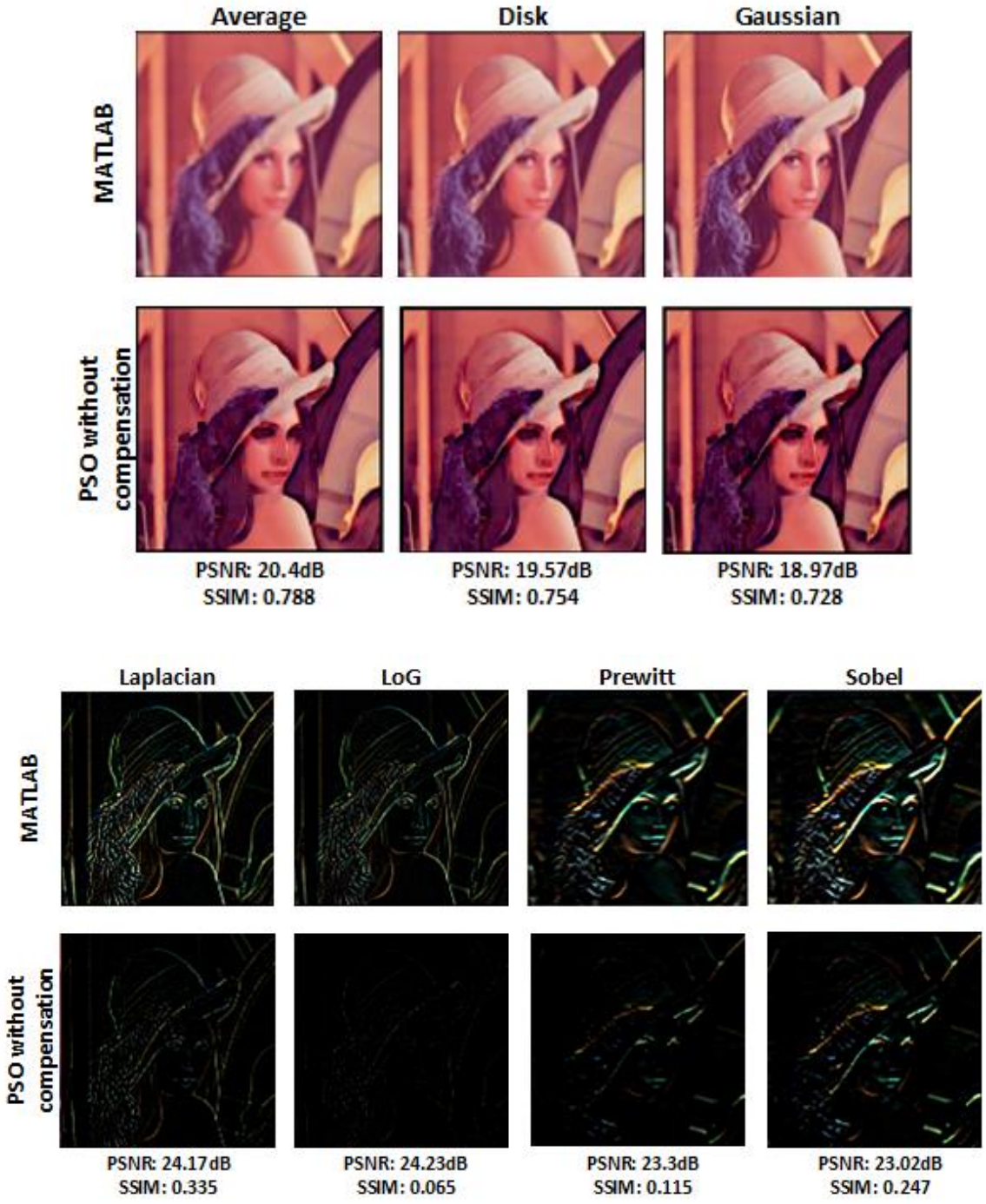
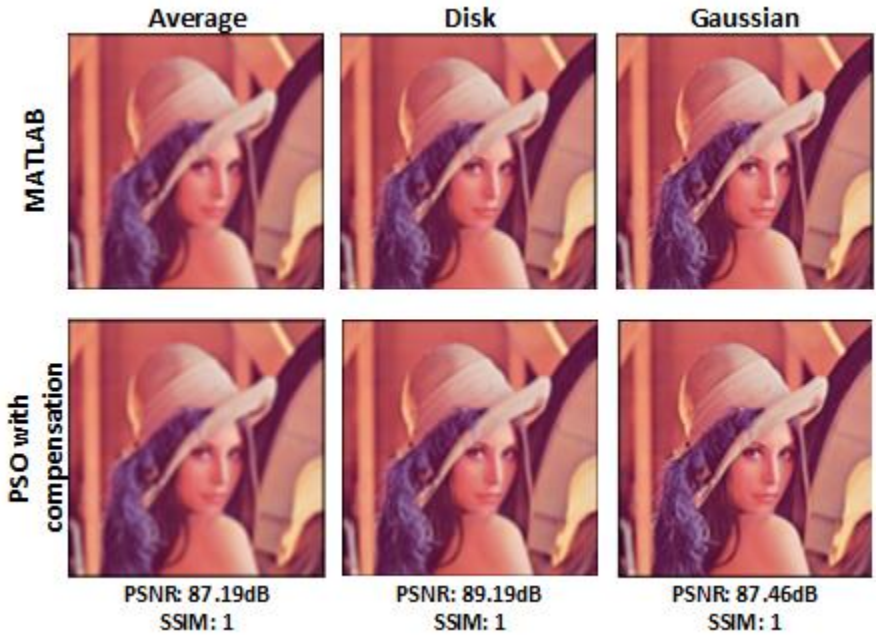


Figure 111: Comparison of image filtering results from MATLAB and PSpice Systems Option without using the compensation algorithm

To implement the compensation algorithm, the seven 3 x 3 kernel values are first mapped into the memristor conductance values and are written into the 9 x 7 memristor-CMOS crossbar array in the PSO environment. This matrix is then multiplied with an input vector consisting of all ones. Each node current is measured and from the resultant node current matrix we compute the effective conductance matrix. By taking the difference between the original and effective conductance matrices we get the ΔG matrix. This ΔG matrix is then added to the effective conductance matrix and this new conductance matrix is written into the memristor-CMOS crossbar array. This process is repeated until the value of ΔG is less than a preset tolerance value. Figure 112 shows the comparison of the image filtering results between MATLAB and PSO with the use of the compensation algorithm. As can be seen from the figure, the PSNR values for all the output filtered images are more than 80dB and the SSIM values for are very close to 1. This shows that the compensation algorithm works extremely well, and it can compensate for wire resistance values, transistor resistance values, voltage drops due on the rows and columns of the crossbar array almost perfectly.



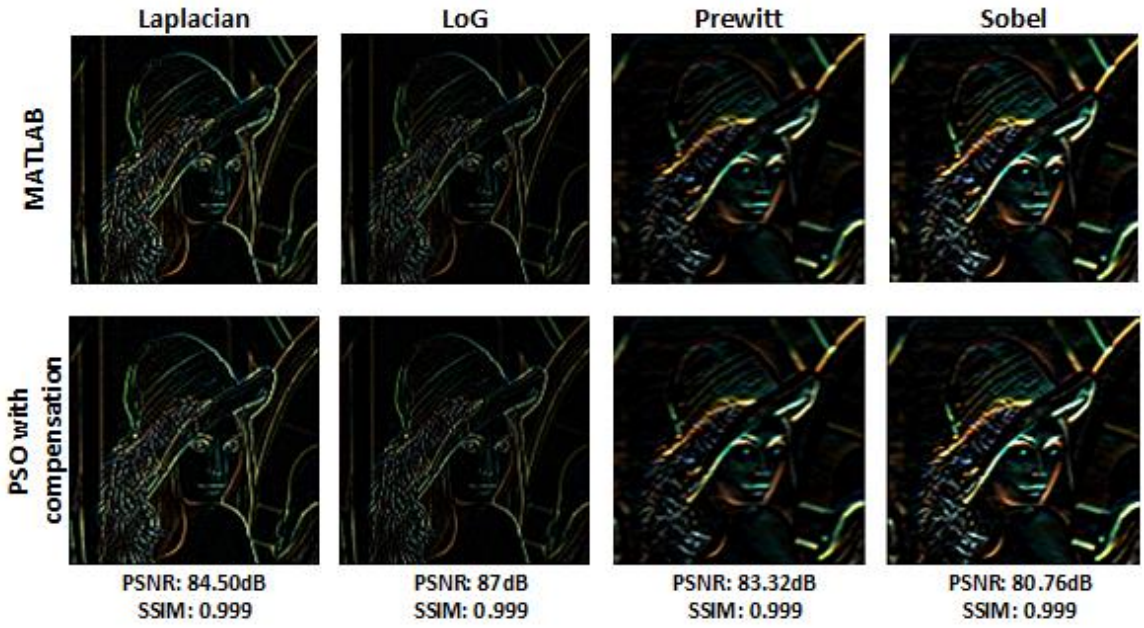


Figure 112: Comparison of image filtering results from MATLAB and PSpice Systems Option after using the compensation algorithm

5.5. Incorporating Memristor Conductance Variation in the PSpice Systems Option Model

In sections 4.5.8 and 4.5.9 we measure the memristor conductance variation during a read operation within a single device and between multiple devices respectively. This is an important measured result which should be incorporated in the analog co-processor model in the PSO environment. A read variation with a Gaussian distribution with a standard deviation of 1% of the target conductance value (mean value) was assumed and incorporated into the model. This variation is more than what was measured in sections 4.5.8 and 4.5.9 and we can treat the simulation results as a worst-case analysis. Table 11 lists the different parameters used in the third analog co-processor model. The major change when compared to the previous model is the incorporation of memristor conductance variation. We demonstrate the effect of including memristor conductance variation through an application simulation of image filtering on the memristor-CMOS crossbar array using PSO simulation environment.

Figure 113 shows the results of performing image filtering on the third analog co-processor model in the PSO environment. As can be seem from the figure, the performance drops when compared to the performance shown in Figure 112 which does not incorporate variation. A gaussian random variation with a standard deviation of 1% can be considered to be a worst-case corner analysis. It can be seen from the figure that the PSNR numbers for all filtering operations are close to 40dB and the SSIM numbers are close to 1. This demonstrates that the performance of the analog co-processor for image processing applications is good even in the presence of memristor cycle to cycle conductance variation.

Table 11: Third Analog Co-Processor model parameters and their values

Analog Co-Processor Model Parameter	Value
Low Resistance State	1e3Ω
High Resistance State	1e4Ω
Number of Bits per Memristor Cell	Analog
Wire Resistance	1Ω
Compensation Algorithm	Used
Memristor Read Variation	Gaussian with S.D. of 1%

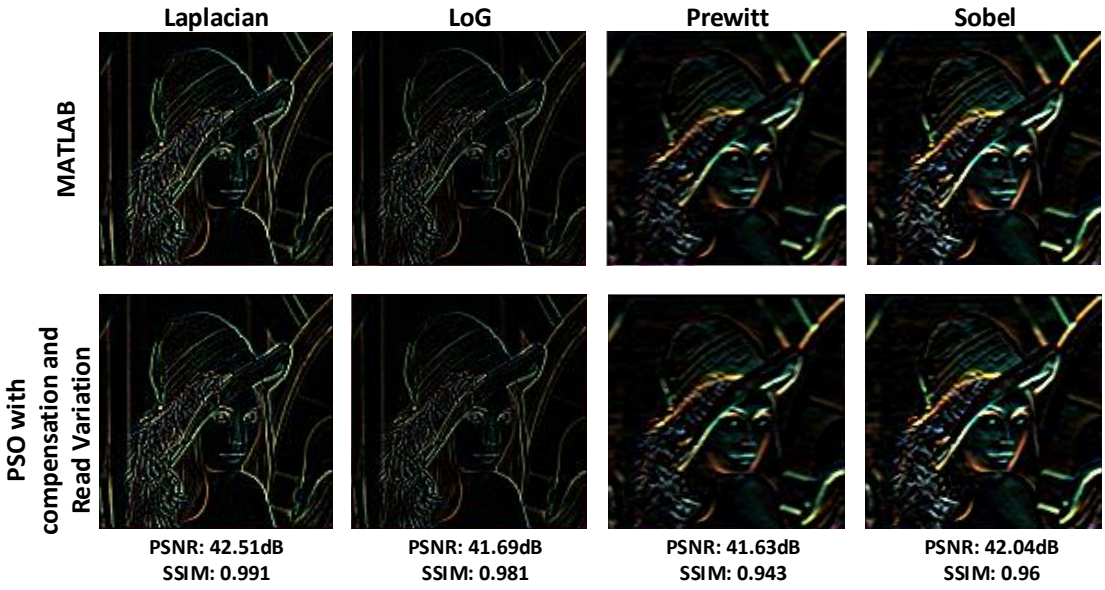


Figure 113: Comparison of image filtering results from MATLAB and PSpice Systems Option after using the compensation algorithm and incorporating cycle to cycle variation

5.6. Floating Point Analog Co-Processor Modeling in the PSpice Systems Option

The aim of the third part of the research was to develop an analog co-processor that can perform 16-bit or half-precision floating point computation. The different versions of the analog co-processor that have been developed in the previous sections can be considered as the building blocks for the final version of the analog co-processor model that can perform 16-bit floating point computation. In order to build the analog co-processor that can handle floating point computation, various sub-blocks had to be built and then these were integrated to realize a fully functional analog co-processor model.

5.6.1. Half-Precision Floating Point Support

The analog co-processor model was designed to handle 16-bit floating point numbers and was being developed in a PSpice Systems Option environment which uses a combination of MATLAB/Simulink and PSpice to perform co-simulations. However, MATLAB does not support

half-precision data formats. In order to handle half-precision datatypes, we developed a MATLAB functional block that can convert any number to and from 16-bit IEEE 754 floating point format. The bit pattern of a 16-bit floating point representation is stored in an unsigned 16-bit integer format in MATLAB. The 16-bit IEEE 754 format is as shown in Figure 47(a) with 1 sign bit, 5 exponent bits biased by 15 and 10 mantissa bits with an implicit leading bit. This block then separates out the sign bit, the exponent value with the bias subtracted and the mantissa value normalized to 1.

5.6.2. Normalizing Block for Floating Point Mantissas

It was explained in section 4.4.1.2 that, to perform floating point addition, the exponents of the two floating point numbers must be equal. To do this, a normalizing block is developed in PSO environment that performs this operation. This block compares the exponents of all the floating point numbers that must be normalized and picks the highest exponent value. All the other floating point numbers are normalized to this exponent by a bit shift operation.

5.6.3. Bit-Slicing and Bit-aggregating Input Blocks for Floating Point Mantissas

It was explained in section 4.4.2 that the analog co-processor can handle high precision computation. However, since the memristor devices can store only a limited amount of data per cell (5-bits in this case), a bit-sliced approach is employed to achieve higher bit precision computation. We have developed a block in PSO environment that can accept 10-bit normalized mantissa values as the input and bit slice it into two 5-bit values known as the MSB bits and LSB bits. After the mantissas have been bit sliced into corresponding MSB and LSB bits, these values have to be aggregated into actual 5-bit numbers before they can be mapped into memristor conductance values or input voltage values. This has also been developed in the PSO environment.

5.6.4. Half-Precision Core and Engine Design of the Analog Co-Processor

In order to handle higher precision computation and support the bit-sliced approach of handling data, we have developed the concept of core and engine as shown in section 4.4.3. For a 16-bit floating point computation with the memristor handling 5-bits of data, the analog co-processor will need 4 cells to perform the required computation. These 4 cells correspond to 1-core and the 1 core is inside an engine of the analog co-processor. Each of these cells are designed as shown in Figure 48 in the PSO environment. The parasitic model of the memristor that was developed in section 4.5.6 is considered for the crossbar array design. Each of the four cells of the analog co-processors handle different bit slices of the input data and are hence named after them. The first cell handles the 5 MSBs of the input matrix and the 5 MSBs of the input vector and hence this cell is called the MSB-MSB cell. The second cell handles the 5 LSBs of the input matrix and the 5 MSBs of the input vector and hence is called the LSB-MSB cell. The third cell handles the 5 MSBs of the input matrix and the 5 LSBs of the input vector and hence is called the MSB-LSB cell. The fourth cell handles the 5 LSBs of the input matrix and the 5 LSBs of the input vector and hence is called the LSB-LSB cell.

5.6.5. Bit-Slicing and Bit-Aggregating Output Blocks for Floating Point Mantissas

The outputs of the analog co-processor are currents in the analog domain. In the actual co-processor design, these outputs will be sampled by ADCs on each column of the memristor-CMOS crossbar array. The bit precision of the ADC for each channel can be calculated as follows:

ADC Bit Precision = Number of Bits Per Memristor + Number of Bits Per Input vector bit slice + Log_2 (Number of Rows in the Memristor-CMOS crossbar array) + 1-bit for the sign

In this design of the analog co-processor we have considered a maximum crossbar array of size 16 x 16 and the memristors can store 5-bits of information. The total bit precision of the

ADC can then be calculated as $5 + 5 + \log_2(16) + 1$ which is equal to 15. For the analog co-processor design in the PSO environment, we simulate the ADC by creating a block that samples the outputs of the column of each cell.

After these values have been sampled by the output bit-slicing block, the outputs from all the four cells must be aggregated to obtain the correct mantissa value. This is achieved by the output bit-aggregating block designed in the PSO environment.

Table 12 lists the parameters used to build the 16-bit floating point model of the analog co-processor in the PSO simulation environment. As can be seen from the table, the major differences between the new model and the previous ones are the restriction imposed on the number of bits per memristor cell, implementation of the complete VMM signal processing chain to enable floating point computation, incorporation of 15-bit ADC blocks in the model and use of the parasitic memristor model developed in section 4.5.6.

Table 12: Fourth Analog Co-Processor model parameters and their values

Analog Co-Processor Model Parameter	Value
Low Resistance State	1e3Ω
High Resistance State	1e4Ω
Number of Bits per Memristor Cell	5-bits
Number of ADC Bits	15-bits
Wire Resistance	1Ω
Compensation Algorithm	Used
Memristor Read Variation	Gaussian with S.D. of 1%
Parasitic Model	Used
Floating Point Computation Enabled	Yes

5.6.6. Application Simulations in the 16-bit Floating Point Analog Co-Processor Model

In this section we test performance of the 16-bit floating point analog co-processor model by demonstrating various image processing applications and solutions to partial differential equations on it.

5.6.6.1. Image Filtering Application Simulation

The first application we demonstrate is image filtering. In this application simulation, the 2D convolution operation is performed on the 16-bit floating point analog co-processor model developed in PSpice Systems Option environment by casting it as a VMM operation. As before we have considered seven different image filtering kernels, each of size 3 x 3. Each kernel matrix is first converted to a column vector of size 9 x 1. Since we have 7 such kernels we get a 9 x 7 matrix of kernel values. These 9 x 7 kernels values are mapped to four cells of the analog co-processor.

Figure 114 shows the comparison of the results of performing linear image filtering operation in MATLAB and the 16-bit floating point model of the analog co-processor in the PSO environment. The first point to be noted in this comparison is that MATLAB is performing computation in double precision and the analog co-processor model is performing the same computation in half precision. As can be seen from the figure, the PSNR performance metrics are close to 40dB or more for all the seven different image filtering operations that have been considered. The SSIM performance metrics are all still very high for all the filtered images with the lowest value being 0.798, which shows that the images are structurally similar as well. We see a slight decrease in performance metric values when compared to the previous model which can mostly be attributed to the decrease in the precision of the analog co-processor from double precision to half precision.

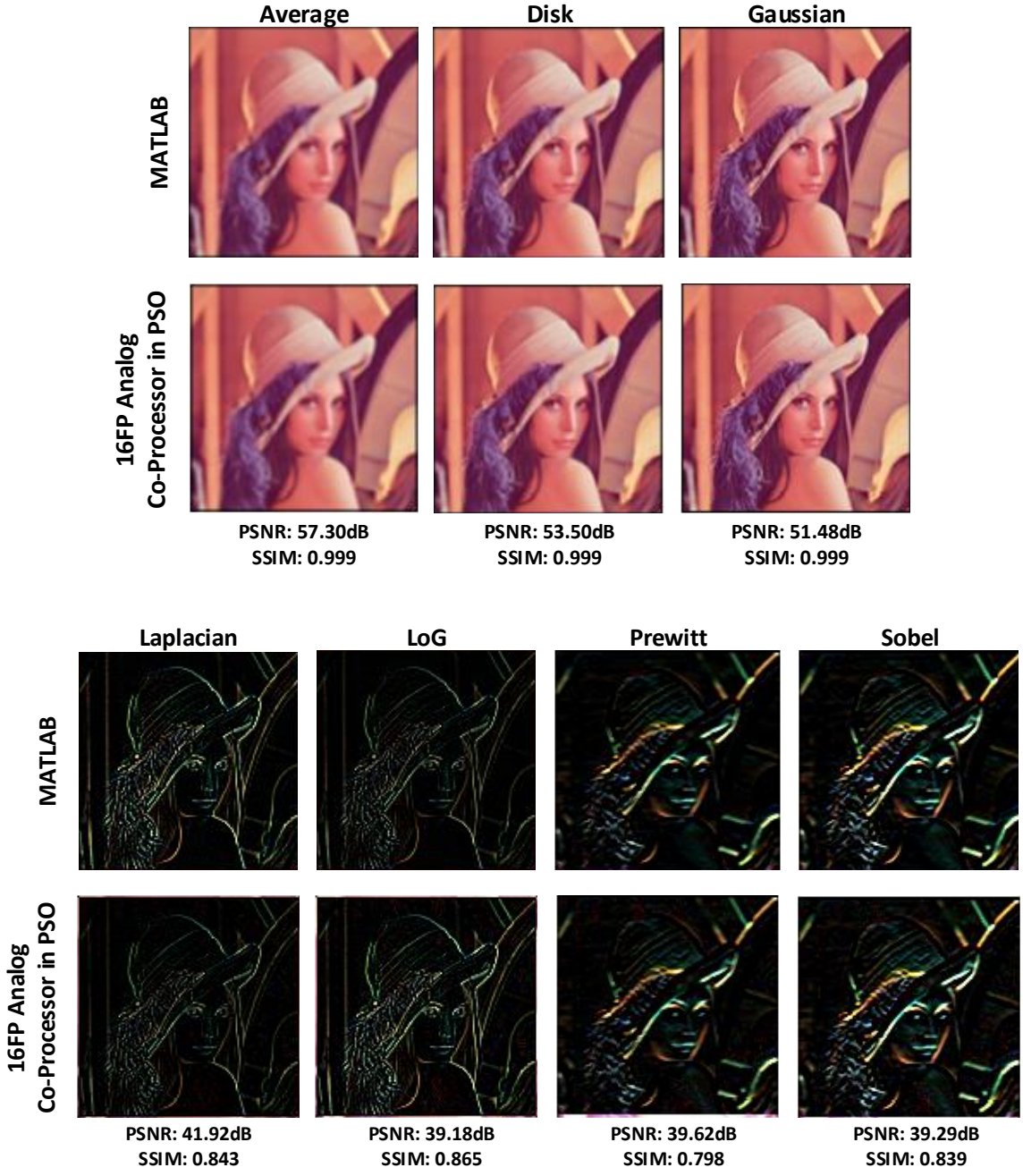


Figure 114: Comparison of image filtering results from MATLAB and 16-bit floating point Analog Co-Processor model in PSpice Systems Option

5.6.6.2. RGB to YUV Color Transformation Simulation Results

In this application simulation, the operation being performed is a general vector-matrix multiplication. The 3 x 3 coefficients from the equations (31), (32) and (33) are mapped to memristor conductance values and then converted into state variable values using the MATLAB/Simulink mapping blocks. These values are then written into a 3 x 3 memristor-CMOS crossbar array. The input images that have to be transformed are converted into vector values and are mapped to memristor-CMOS crossbar array read voltages using the mapping block developed in MATLAB/Simulink. These voltages are input into the memristor-CMOS crossbar array to perform VMM operation in PSpice Systems Option co-simulation environment. The output currents are sampled and brought back to the MATLAB/Simulink environment where they are inverse mapped to convert them to actual pixel intensity values to be displayed as output transformed images.

Figure 115 shows the input RGB image and the *Y*, *U* and *V* components generated from both MATLAB and the 16-bit floating point model of the analog co-processor in the PSO environment. Qualitatively both the MATLAB and PSpice Systems Option results look similar. The PSNR for the analog Y component is 70.83dB, analog *U* component is 71.97dB and analog *V* component is 71.2dB. The SSIM values for the Y, U and V components are 0.98, 0.707 and 0.682 respectively. These PSNRs and SSIMs were measured by taking the MATLAB simulated outputs as true images. We see a slight decrease in performance metric values when compared to the first model which can mostly be attributed to the decrease in the precision of the analog co-processor from double precision to half precision.

The high values of the PSNRs and SSIMs indicate that the analog co-processor can be used very efficiently for performing color image transformations which are used very frequently in emerging machine vision applications.

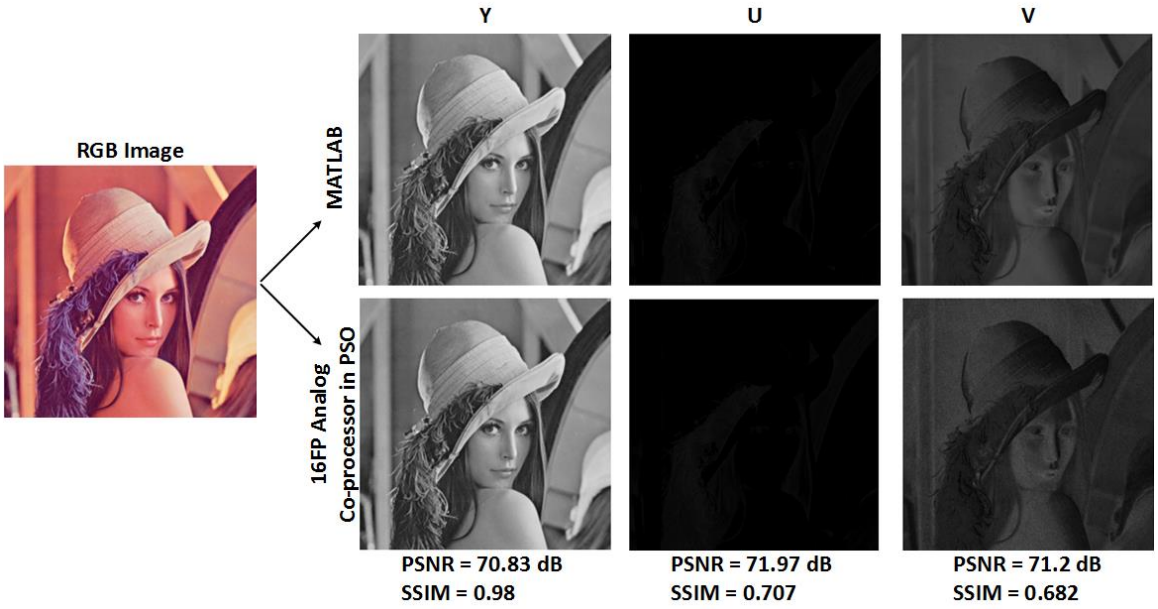


Figure 115: Comparison of RGB to YUV color transformation on the Lena image in MATLAB and 16bit analog co-processor model in PSO environment

5.6.6.3. Solution to 2D Poisson Equation

For this application simulation, we considered the solution of a 2D Poisson PDE where three point-source terms were present. This is equivalent to finding the electric potential that results from three point charges distributed on the plane. It is also equivalent to finding the temperature in a large surface that has three point heaters/coolers. The resolution of the problem considered was 128 x 128. The size of the mesh was fixed at 16 x 16. The total number of meshes would be 64.

Figure 116 shows the solution to the PDE obtained using the 16-bit floating point analog co-processor model. The scale on the right indicates the intensity of the electrostatic field due to the three point charges. Figure 117 shows the residual plot which indicates the number of iterations it takes the analog co-processor to converge to a solution with an error tolerance of 10^{-10}. For a problem size of 128 x 128 and a Green's function mesh size of 16 x 16 the analog co-processor takes 355 iterations to converge with an average step error of 3.45% when compared

to an ideal MATLAB simulation. A 16-bit MATLAB simulation of the same PDE takes 353 iterations to converge to a solution. It was shown in section 4.4.5 that the theoretical performance of the analog processor is 15x faster than the projected 5nm GPU for a 32-bit operation. Assuming this performance, the analog co-processor converges to the solution of a Poisson PDE 14.9 times faster than a projected 5nm GPU.

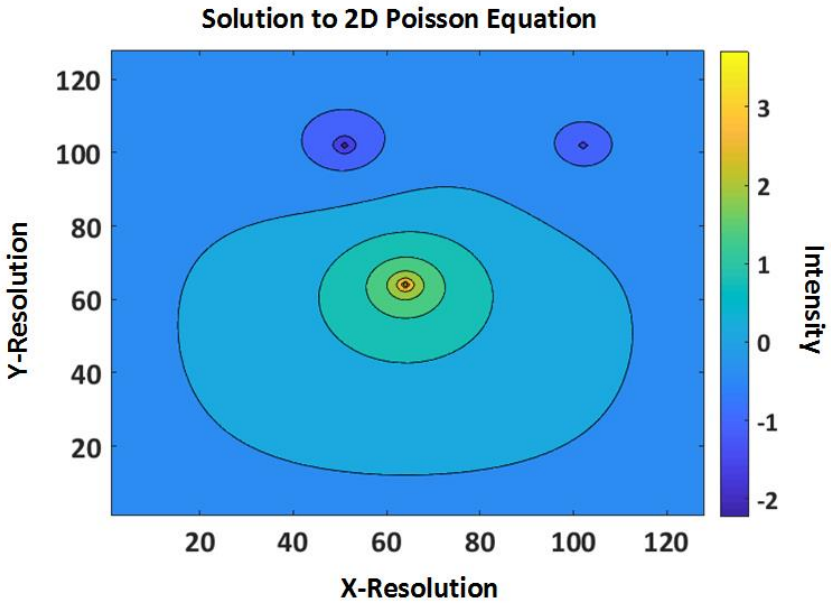


Figure 116: Solution to 2D Poisson Partial Differential Equation with a resolution of 128 x 128 obtained using the 16-bit floating point model of the analog co-processor

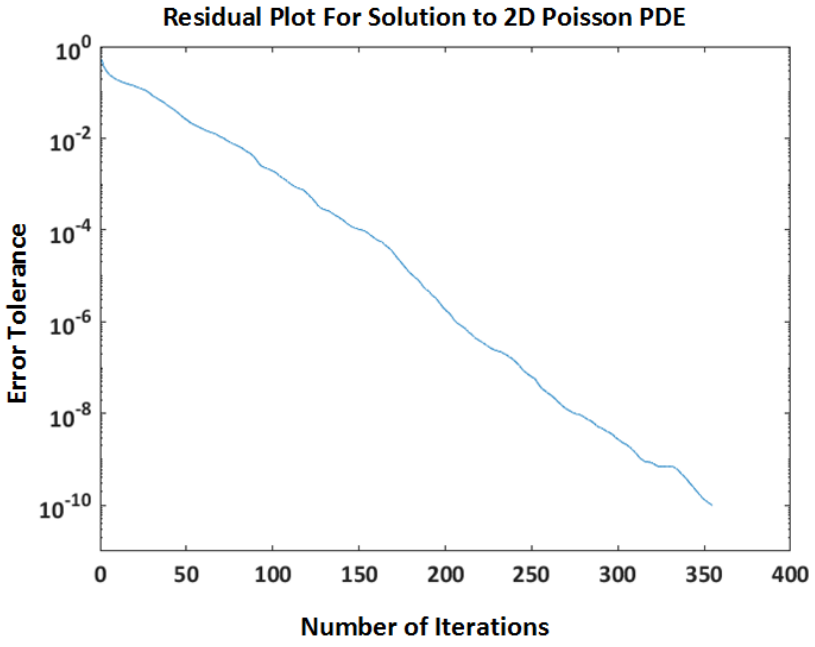


Figure 117: Residual plot of solution to 2D Poisson equation with a resolution of 128 x 128 obtained using the 16-bit floating point model of the analog co-processor

5.6.6.4. Solution to 2D Laplace Equation

For this application simulation, we considered a Laplace PDE with a resolution of 128 x 128 and a Green's function mesh size of 16 x 16. The total number of meshes were 64. Figure 118 shows the solution to the PDE obtained using the 16-bit floating point analog co-processor model. Figure 118 show the solution to the 2D Laplace PDE obtained using the 16-bit floating point analog co-processor model. Figure 119 shows the residual plot for the solution to the 2D Laplace equation which gives the number of iterations required for the analog co-processor to converge to a solution with an error tolerance of 10^{-10}. The analog co-processor converges to a solution in 377 iterations with an average step error of 18.54% when compared to the ideal 16-bit MATLAB simulation of the same PDE which converges in 346 iterations. The error in the preconditioner for the solution of the Laplace PDE is higher than it was before. This can be attributed to the reduction

in precision from double precision to half precision. As it can be seen from the results, even with a larger error during pre-conditioning, the analog co-processor converges to the correct solution within a reasonable number of iterations. It was shown that the theoretical performance of the analog processor is 15x faster than the projected 5nm GPU performance for a 32-bit operation. Assuming this performance, the analog co-processor converges to the solution of a Laplace PDE 13.76 times faster than a projected 5nm GPU.

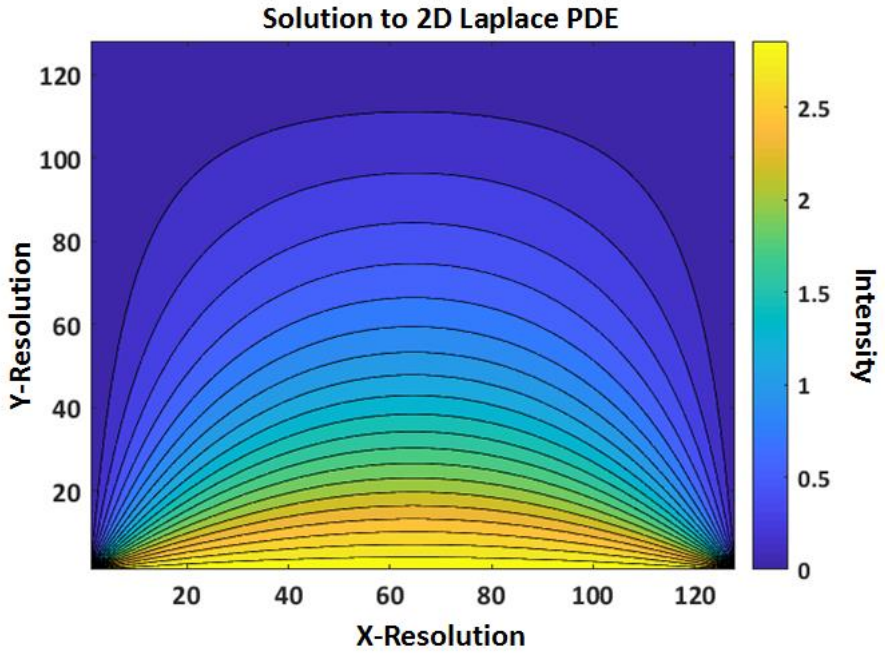


Figure 118: Solution to 2D Laplace Partial Differential Equation with a resolution of 128 x 128 obtained using the 16-bit floating point model of the analog co-processor

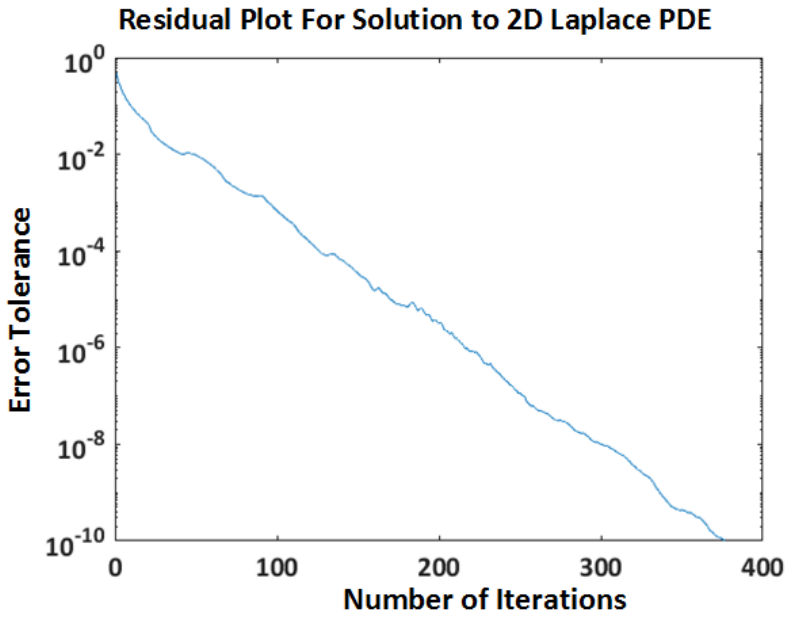


Figure 119: Residual plot of solution to 2D Laplace equation with a resolution of 128 x 128 obtained using the 16-bit floating point model of the analog co-processor

4.13. Conclusion

In this part of the dissertation we develop the Memristor-CMOS analog co-processor model in PSpice Systems Option environment. The model is developed in steps and at each step, the performance of the model is verified by simulating applications on the analog co-processor model. Each version of the analog co-processor model developed is an improvement over the previous version. The final version of the analog co-processor can perform half precision floating point computation.

Two major applications are simulated on the analog co-processor model. The first set of applications that are simulated are related to image processing and the second set of applications that are simulated are related to the solution of PDEs. The application simulations demonstrate that the memristor-CMOS analog co-processor can be used as an accelerator for some of the emerging high-performance computing applications.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The aim of this dissertation was to demonstrate that analog signal processing is a viable alternative to digital processing and we believe we have successfully demonstrated this through three major parts of the dissertation.

In the first part of the dissertation it was demonstrated that highly computational stereo correspondence algorithms can be accelerated through slight modifications. The modifications to these algorithms reduced the computational complexity by a factor of 'N' and made these algorithms efficiently implementable in the analog domain.

In the second part of the dissertation, a prototype of a CMOS based analog processor was designed and implemented using COTS components. A variety of simulations and measured results showed that the analog processor can be used to perform low-level and mid-level image processing tasks which are used frequently in Machine vision algorithms. An integrated CMOS based analog processor design was proposed and it was demonstrated that such a processor could outperform existing state-of-art digital and analog processors.

In the third part of the dissertation, a Memristor-CMOS analog co-processor that can perform floating point computations was designed and a model of this analog co-processor was developed in a PSpice Systems Option environment. Through a variety of measurements and application simulations, it was shown that the analog co-processor can accelerate high-performance computing applications.

As part of the future work this memristor-CMOS based analog co-processor can be extended to include read and write circuitry in PSPICE environment. The models of the memristor-CMOS crossbar array can also be improved through more fabrication and measurements. A

prototype of the co-processor can also be built using COTS components for application demonstrations. This will finally lead to an integrated circuit design of such an analog co-processor.

BIBLIOGRAPHY

[1] "Taking Advantage of Spatial Redundancy", *Technical Paper*, Dept. of Computer Science, Dartmouth College

[2] Adhikari, G. et al. (2012) "Fast normalized cross-correlation with early elimination condition", *IEEE Transaction, ICRTIT 2012*, pp136-140.

[3] Alan Stocker, "Analog VLSI focal-plane array with dynamic connections for the estimation of piecewise-smooth optical flow," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 963-973, May 2004.

[4] Alma Delic et al., "Programmable State-Variable Filter with Wide Frequency and Q Tuning Range", *International Journal of Electronics and Electrical Engineering,* Vol.3, No.3, pp. 207-211, June. 2015.

[5] Amin Shameli et al., "A Novel Ultra-Low Power Low Noise Amplifier using differential inductor feedback", *IEEE European Solid-State Conference,* pp. 352-355, Sept. 2006.

[6] Analog Devices, "AD4817-1 Low-Noise, 1GHz, FastFET Op Amps", *AD4817 Datasheet*

[7] Analog Devices, "AD8336 Wide Bandwidth, DC-coupled VGA", *AD8336 Datasheet*.

[8] Analog Devices, "ADG601/602 SPST Switches", *ADG601/602 Datasheet*

[9] Analog Devices, "ADL5391 DC to 2GHz, Multiplier", *ADL5391 Datasheet*

[10] Analog Devices, Available; http://www.analog.com/en/products/switches-multiplexers/analog-switches-multiplexers/adg901.html

[11] ARM Community, Available: https://community.arm.com/processors/b/blog/posts/introducing-cortex a32-arm-s-smallest-lowest-power-armv8-a-processor-for-next generation-32-bit-embedded-applications

[12] Athreyas, N., Gupta, D., and Gupta, J., "Analog signal processing solution for machine vision applications", *Journal of Real-Time Image Processing* [online] (Feb 2017), 1-22. Available: https://link.springer.com/article/10.1007/s11554-017-0669-4

[13] Bojnordi, M. and Ipek, E., “Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning”, in *Proc. HPCA*, Barcelona, Spain, 2016, 1-13.

[14] Boser, B. E., Sackinger, E., Bromley, J., Le Cun, Y., and Jackel, L. D. “An Analog Neural Network Processor with Programmable Topology”, *IEEE Journal of Solid-State Circuits*, 26, 12(Dec. 1991), 2017-2025.

[15] Briechle, Kai, & Uwe D. Hanebeck, (2001) "Template matching using fast normalized cross-correlation", *Aerospace/Defense Sensing, Simulation, and Controls. International Society for Optics and Photonics.*

[16] Brown, L, (1992) “A survey of image registration techniques”, *ACM Journal* (*CSUR*), Vol. 24, Issue 4, pp325-376.

[17] Cadence, Available: http://www.pspice.com/technology/pspice-systems-option.

[18] Chin Yin, et al., “A 0.5V 34.4uW 14.28kfps 105dB smart image sensor with array level analog signal processing”, *IEEE Solid-State Circuits Conference,* pp. 97-100, Nov. 2013

[19] Choi, S., Sheridan, P., and Lu, W. D., “Data clustering using memristor networks”, *Scientific Reports*, 5, 10492 (May 2015).

[20] Christopher Soell., et al., "A CMOS image sensor with analog pre-processing capability suitable for smart camera applications”, *IEEE ISPACS,* pp. 279-284, Nov. 2015.

[21] Chua, Leon O., “Memristor-The Missing Circuit Element”*, IEEE Trans. On Circuit Theory*, 18, 5(Sept. 1971).

[22] Chua, Leon O., “The Fourth Element”*, Proc. of IEEE,* 100, 6(Apr. 2012).

[23] Chung, Fan, and Yau, S-T., “Discrete Green's functions”, *Journal of Combinatorial Theory,* 91, 1-2(July 2000), 191-214.

[24] Craig Schlottmann., et al., "Vector matrix multiplier on field programmable analog array," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1522-1525, March 2010.

[25] De Simone, F., Ticca, D., Dufaux, F., Ansorge, M., and Ebrahimi, T., “A comparative study of color image compression standards using perceptually driven quality metrics”, in *SPIE Optics and Photonics, Applications of Digital Image Processing,* San Diego, CA, Aug. 2008.

[26] Devereux, “Limiting of YUV Digital Video Signals”, *BBC Research Department,* Dec. 1987.

[27] Dosselmann and Yang, “A comprehensive assessment of the structural similarity index”, *Signal, Image and Video Processing*, Vol.5, Issue 1, pp. 81-91, Nov. 2009.

[28] Eugenio Culurciello, “Enabling machines to perceive the world”, *Teradeep Inc.,* 2014.

[29] Federico Corradi, et al., "Decision making and perceptual bistability in spike-based neuromorphic VLSI systems", *IEEE International Symposium on Circuits and Systems* (*ISCAS*), Lisbon, 2015, pp. 2708-2711.

[30] Francesca De Simone, et al., “A comparative study of color image compression standards using perceptually driven quality metrics”, *Optical Engineering Applications*, *SPIE,* Aug 2008.

[31] G. Linan, et al. “A 0.5µm CMOS 10^6 transistors analog programmable array processor for real-time image processing”, *IEEE Solid-State circuits conference*, pp 358-361, Sept. 1999.

[32] Genov, R. and Cauwenberghs, G., “Charge-Mode Parallel Architecture for Vector-Matrix Multiplication”, in *Trans. IEEE Circuits and Systems II*, 48, 10(Oct. 2001), 930-936.

[33] Genov, R. and Cauwenberghs, G., “Kerneltron: support vector "machine" in silicon”, *IEEE Trans. on Neural Networks*, 14, 5(Nov. 2003), 1426-1434.

[34] Giacomo Indiveri, et al., "A reconfigurable neuromorphic VLSI multi-chip system applied to visual motion computation", *Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, (MicroNeuro '99)., Granada, 1999, pp. 37-44.

[35] Gonzalez, R., and Woods, R., “Digital Image Processing”, *Upper Saddle River NJ: Prentice Hall*, 2002.

[36] Guillermo Botella, et al., "Bio-inspired robust optical flow processor system for VLSI implementation", *Electronics letters*, Vol.45, Issue 25, pp. 1304-1305, 2009.

[37] Guillermo Botella, et al., "Robust bioinspired architecture for optical-flow computation", *IEEE Trans. VLSI Syst*ems. Vol.18, Issue 4, pp. 616-629, 2010.

[38] Gupta, N, “A VLSI architecture for Image Registration in Real Time”, *IEEE Transactions on VLSI systems*, Vol. 15, Issue 9, pp981-989, 2007.

[39] H. Tang, “Study of design for reliability of RF and analog integrated circuits”, *PhD. Dissertation*, University of Central Florida, Orlando, Florida, 2012.

[40] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, Mark A., and Dally, William J., "EIE: efficient inference engine on compressed deep neural network", In *Proc. ISCA*, Seoul, South Korea, 2016, 243-254.

[41] Harpe, P., Zhang, Y., Dolmans, G., Philips, K., and Groot, H. D., "A 7-to-10b 0-to-4MS/s flexible SAR ADC with 6.5-to-16fJ/conversion-step", in *ISSCC*, San Francisco. CA, 2012, 472-474.

[42] Hestenes, M. R., and Stiefel, E., "Methods of conjugate gradients for solving linear systems", *Journal of Research of the National Bureau of Standards*, Vol. 49, no. 1, pp. 409-436, May 1952.

[43] Hirschmüller, H. and Scharstein, D., "Evaluation of cost functions for stereo matching", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR), 2007.

[44] Hu, J., Xue, C. J., Zhuge, Q., Tseng, W-C., and Sha, E. H-M. Towards energy efficient hybrid on-chip scratch pad memory with non-volatile memory. in *DATE,* Grenoble, France, 2011, 1-6.

[45] Hu, M., Strachan, J. P., Li, Z., Grafals, E. M., Davila, N., Graves, C., Lam, S., Ge, N., Williams, R. S., and Yang, J., "Dot-Product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Accelerate Matrix-Vector Multiplication", in *Procs. of DAC-53*, Austin, TX, 2016.

[46] Intel, Available: https://www.intelnervana.com/neon/

[47] Ion Vornicu et al. "Image Processing using a CMOS analog parallel architecture", *IEEE Proceedings on CAS*, Vol. 2, Oct. 2010.

[48] J. S. Kong et. al., "A 160x120 Edge Detection Vision Chip for Neuromorphic Systems Using Logarithmic Active Pixel Sensor with Low Power Dissipation", *ICONIP*, 2007.

[49] J.L.Wyatt et al."The MIT Vision chip project: analog VLSI systems for fast image acquisition and early vision processing", *IEEE international conference on Robotics and Automation*, April 1991.

[50] J.-O. Klein, et al., "Low power image processing: Analog versus Digital comparison", *IEEE International Workshop on Computer Architecture for Machine Perception,* pp. 111-115, July. 2005.

[51] Jain, Anil K., *Fundamentals of Digital Image Processing,* Englewood Cliffs, NJ, Prentice Hall, 1989, pp. 150-153.

[52] Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates et al., “In-datacenter performance analysis of a tensor processing unit”, in *ISCA*, Toronto, Canada, June 2017.

[53] Jung-Hwan Kim, et al., “A Low Power Analog CMOS Vision Chip for Edge Detection Using Electronic Switches”, *ETRI Journal*, Vol. 27, Issue 5, Oct. 2005.

[54] K. Shimonomura, et al., "Neuromorphic binocular vision system for real-time disparity estimation," *IEEE International Conference on Robotics and Automation*, pp. 4867-4872, Roma, 2007.

[55] Kalin Ovtcharov, et al., “Accelerating Deep Convolutional Neural Networks Using Specialized Hardware”, *Microsoft Research*, Feb. 2015

[56] Khaleghi, B et al. “A new Miniaturized Embedded Stereo-Vision System (MESVS-I)”, *CRV*, pp26-33, 2008.

[57] Klaus, A., Sormann, M. and Karner, K. “Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure”, ICPR, 2006.

[58] Krizhevsky, et al., “ImageNet Classification with Deep Convolutional Neural Networks”, *Advances in Neural Information Processing Systems*, 2012.

[59] Kub, F., Moon, K., Mack, I., and Long, F. Programmable analog vector-matrix multipliers, *IEEE Journal of Solid-State Circuits*, 25, 1(Feb 1990), 207-214.

[60] Lewis, Dave. SerDes Architectures and Applications, *DesignCon*, Santa Clara, CA, Feb. 2004

[61] Lewis, J.P, (1995) “Fast Normalized Cross-correlation”, *Industrial Light & Magic.*

[62] Lin, Wei-Te, Huang, H-Yi., and Kuo, Tai-H, “A 12-bit 40 nm DAC Achieving SFDR> 70 dB at 1.6 GS/s and IMD<–61dB at 2.8 GS/s With DEMDRZ Technique”, *IEEE Journal of Solid-State Circuits*, 49, 3(Feb. 2014), 708-717.

[63] Mahdi Parvizi, et al., “An Ultra-Low-Power Wideband Inductorless CMOS LNA With Tunable Active Shunt-Feedback,” *IEEE Transaction on Microwave Theory and Techniques*, Vol. 64, No. 6, June 2016.

[64] Mathworks, Available: https://www.mathworks.com/help/images/ref/fspecial.html

[65] Minicircuit, “RLP-70+ Low Pass Filter”, *RLP-70+ Datasheet*.

[66] Moon, K. K., Kub, F. J., and Mack, I. A., “Random address 32 times; 32 programmable analog vector-matrix multiplier for artificial neural networks”, in *Procs. IEEE Custom Integrated Circuits Conference*, Boston, MA, May 1990, 26.7/1-26.7/4.

[67] Movidius, “Myriad 2 MA2x5x Vision Processor VPU Product Brief”, 2016

[68] Movidius, “Software Programmable Media Processor”, Aug. 2011.

[69] Mrigank Sharad et al., "Ultra-low energy analog image processing using spin based neurons,” IEEE/ACM International Symposium on Nanoscale Architectures, pp. 211-217, July 2012.

[70] Nvidia, “GPU-based deep learning inference: A performance and Power Analysis”, *Nvidia Whitepaper,* Nov. 2015.

[71] Nvidia, “NVIDIA Tesla P100”, *Nvidia Whitepaper*.

[72] Nvidia, Available: https://www.nvidia.com/en-us/data-center/volta-gpu-architecture

[73] P Kinget et al. “A Programmable analog cellular neural network CMOS chip for high speed image processing," *IEEE Journal of Solid-State Circuits*, Vol. 30, Issue 3, pp. 235-243, Mar. 1995.

[74] P. Dudek et al., “A general-purpose processor-per-pixel analog SIMD vision chip”, *IEEE Transaction on Circuits and Systems*, Vol.52, Issue 1, pp. 13-20, 2005.

[75] Pan, X., and Graeb, H., “Reliability optimization of analog integrated circuits considering the tradeoff between lifetime and area”, *ICMAT*, *Elsevier*, 52, 8(Oct. 2011), 1559-1564.

[76] Parvizi, Mahdi, Allidina, K., and El-Gamal, M. N., “An ultra-low-power wideband inductorless CMOS LNA with tunable active shunt-feedback”, *IEEE Transactions on Microwave Theory and Technique,* 64, 6(May 2016), 1843-1853.

[77] Pennebaker, William B., and Mitchell, Joan L., *JPEG: Still Image Data Compression Standard.* Van Nostrand Reinhold, 1993.

[78] Pieter Harpe et al., “A 7-to-10 bit 0-to-4 MS/s flexible SAR ADC with 6.5-to-16fJ/conversion step”, *IEEE Journal of Solid-State Circuits,* pp. 472-474, Feb. 2012.

[79] R Carmona et al. “A 0.5µm CMOS CNN Analog Random Access Memory Chip for Massive Image Processing,” *IEEE International workshop on CNN and their Applications*, April 1998.

[80] R. Gotarredona, et al., "A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 12, pp. 2548-2566, Dec. 2006.

[81] Radhamani and R, Keshaveni, (2012) “FPGA implementation of efficient and High speed template matching module”, *IJRTE*, Vol. 2, Issue 2.

[82] Rafael Gonzalez and Richard Woods, “Digital Image Processing”, *Upper Saddle River NJ: Prentice Hall*, 2002.

[83] Roma, N et al. (2002) “A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach”, *INESC*, Portugal.

[84] Rzeszutek, R., Dong Tian and Vetro, A. “Disparity estimation of misaligned images in a scanline optimization framework”, *ICASSP*, 2013.

[85] S. Espejo., et al. "A 0.8μm CMOS Programmable Analog-Array-Processing Vision Chip with Local Logic and Image-Memory", *European IEEE Solid-State Circuits Conference*, pp. 280-283, Sept. 1996.

[86] S. Winkler, “Digital Video Quality: Vision Models and Metrics”, *John Wiley & Sons Ltd*., West Sussex, 2005.

[87] San Yong, Y., and Hon, H. W. (2008) “Disparity estimation for objects of interest”, *World Academy of Science*, Engineering and Technology, 43.

[88] Sasanka Potluri et al., “CNN based high performance computing for real-time image processing on GPU,” *IEEE Proceedings of the Joint INDS and ISTET*, pp. 1-7, July. 2011.

[89] Scharstein, D. and Szeliski, R. (2002) “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”, *International Journal of Computer Vision*, pp 7-42.

[90] Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., and Srikumar, V., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars”, in *Procs ISCA*, Seoul, South Korea, 2016, 14-26.

[91] Sheridan, P. M., Du, C., and Lu, W. D., “Feature extraction using memristor networks”, *IEEE Trans. on Neural Networks and Learning Systems*, 27, 11 (Nov. 2016), 1-10.

[92] Sheridan, P., Ma, W., and Lu, W., “Pattern recognition with memristor networks”, in *ISCAS*, Melbourne, Australia, 2014, 1078-1081.

[93] Shih-Chii Liu, "A neuromorphic aVLSI model of global motion processing in the fly," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 12, pp. 1458-1467, Dec 2000.

[94] Sohmers, T., EE380: Computer Systems Colloquium Seminar, The REX Neo Architecture: An energy efficient new processor architecture for HPC, DSP, Machine Learning, and more, Feb 2017, Available: https://www.youtube.com/watch?v=ki6jVXZM2XU.

[95] Stam, Jos., “Stable fluids”, In *Proc. SIGGRAPH*, 1999, 121-128.

[96] Strachan, J. P., Torrezan, Antonio C., Miao, F., Pickett, Matthew D., Yang, J. Joshua, Yi, W., Medeiros-Ribeiro, Gilberto, and Williams, R. Stanley., “State dynamics and modeling of tantalum oxide memristors”, *IEEE Transactions on Electron Devices*, 60, 7(July 2013), 2194-2202.

[97] Szeliski, R, “Image Alignment and Stitching: A tutorial”, *Technical Report*, Microsoft Research, Microsoft Corporation, MSR-TR-2004-92, 2004.

[98] T. F. Chan, G. H. Golub, and R. J. LeVeque, “Algorithms for computing the sample variance: Analysis and recommendations”, *The American Statistician*, 37, pp. 242–247, 1983

[99] Tang, H., Study of design for reliability of RF and analog integrated circuits. PhD. Dissertation, Dept. Electrical Eng., University of Central Florida, Orlando, Florida, 2012.

[100] Tao, L., Liu, S., Li, L., Wang, Y., Zhang, S., Chen, T., Xu, Z., Temam, O., and Chen, Y., “DaDianNao: a neural network supercomputer”, *IEEE Trans. on Computers*, 66, 1(May 2016), 73-88.

[101] Tetsuya Yagi and K. Shimonomura, "Silicon primary visual cortex designed with a mixed analog-digital architecture," *International Joint Conference on Neural Networks*, pp. 2723-2728, Orlando, FL, 2007.

[102] Texas Instruments, “LMH6552 1.5GHz Fully Differential Amplifier”, *LMH6552 Datasheet*, Apr. 2007 [Revised Jan. 2015].

[103] Tsai, Du-Ming and Chien-Ta Lin, “Fast normalized cross-correlation for defect detection”, *Pattern Recognition Letters* 24.15, pp2625-2631, 2003.

[104] Ttofis, C., and Theocharides, T. “Towards accurate hardware stereo correspondence: A real-time FPGA implementation of a segmentation-based adaptive support weight algorithm”, *In Proceedings of the Conference on Design*, Automation and Test in Europe, pp703-708.

[105] Tyson Hall, “Field-Programmable Analog Arrays: A Floating-Gate Approach”, *PhD. Dissertation,* Georgia Institute of Technology, Atlanta, GA, July 2004.

[106] Valle et al., "An analog VLSI neural network for real-time image processing in industrial applications," *Seventh IEEE ASIC Conference*, pp. 37-40, Sept. 1994.

[107] Vatanjou, Asghar, A., Ytterdal, T., and Aunet, S., “Energy efficient sub/near-threshold ripple-carry adder in standard 65 nm CMOS”, In *ASQED,* Kula Lumpur, Malaysia, 2015, 7-12.

[108] W. Jendernalik, et al., “Analog CMOS processor for early vision processing with highly reduced power consumption”, *20th European Conference on Circuit Theory and Design*, pp. 745-748, 2011

[109] W. Jendernalik, et al., “CMOS realization of analogue processor for early vision processing”, *Bulletin of the Polish Academy of Sciences: Technical Science,* Vol. 59, Issue 2, pp. 141–147, Aug. 2011

[110] Wei Miao et al. “A Programmable SIMD Vision Chip for Real-Time Vision Applications, *IEEE Journal of Solid-State Circuits*, Vol. 43, Issue 6, pp. 1470-1479, June 2008.

[111] Wei Yi et al., “Quantized conductance coincides with state instability and excess noise in tantalum oxide memristors”, *Nature Communications*, Oct. 2017.

[112] We-Te Lin et al., “A 12-bit 40 nm DAC Achieving SFDR > 70 dB at1.6 GSPS and IMD < -61 dB at 2.8 GSPS with DEMDRZ Technique”, *IEEE Journal of Solid-State Circuits,* pp. 708-717, Mar. 2014.

[113] Winkler, S., Digital Video Quality: Vision Models and Metrics. *John Wiley & Sons Ltd*., West Sussex, 2005

[114] Xiaotao Wang and Xingbo Wang, "FPGA Based Parallel Architectures for Normalized Cross-Correlation”, *1st International Conference Information Science and Engineering* (*ICISE*), pp225-229, 2009.

[115] Xilinx, “7 Series FPGAs overview”, Sept. 2016, DS180 (v2.0)

[116] Xilinx, “Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics”, *Artix-7 Datasheet*, Nov. 2015.

[117] Xin Pan and Helmut Graeb, “Reliability optimization of analog integrated circuits considering the tradeoff between lifetime and area”, *ICMAT*, *Elsevier*, Vol. 52, Issue 8, pp. 1559-1564, Oct. 2011

[118] Xinyu Wu, et al., "Homogeneous Spiking Neuromorphic System for Real-World Pattern Recognition," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 254-266, June 2015.

[119] Y. Papananos."Feasibility of analogue computation for image processing application," *IEEE Proceedings- Circuits, Devices and Systems,* Vol.136, Issue 1, pp. 9-13, Feb 1989.

[120] Yang, Byung-Do., “Low-power and area-efficient shift register using pulsed latches”, *IEEE Trans. on Circuits and Systems I: Regular Papers*, 62, 6(May 2015), 1564-1571.

[121] Yang, J. Joshua, Strukov, Dmitri B., and Stewart, Duncan R., “Memristive devices for computing”, *Nature nanotechnology*, 8, 1(Dec 2012), 13-24.