AN ASIC CHIP DESIGN WITH ADVANCED IMAGE CORRECTION ARCHITECTURES FOR CMOS INFRARED IMAGERS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÖRKEM CİLBİR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

JUNE 2024

Approval of the thesis:

AN ASIC CHIP DESIGN WITH ADVANCED IMAGE CORRECTION ARCHITECTURES FOR CMOS INFRARED IMAGERS

submitted by **GÖRKEM CİLBİR** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, **Graduate School of Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronics Eng.** _____

Prof. Dr. Tayfun Akın
Supervisor, **Electrical and Electronics Eng., METU** _____

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Eng., METU _____

Prof. Dr. Tayfun Akın
Electrical and Electronics Eng., METU _____

Prof. Dr. Ali Bozbey
Electrical and Electronics Eng., TOBB ETÜ _____

Assoc. Prof. Dinçer Gökcen
Electrical and Electronics Eng., Hacettepe University _____

Assist. Prof. Hasan Uluşan
Electrical and Electronics Eng., METU _____

Date: 07.06.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Görkem Cilbir

Signature :

ABSTRACT

AN ASIC CHIP DESIGN WITH ADVANCED IMAGE CORRECTION ARCHITECTURES FOR CMOS INFRARED IMAGERS

Cilbir, Görkem
Master of Science, Electrical and Electronic Engineering
Supervisor: Prof. Dr. Tayfun Akın

June 2024, 110 pages

This thesis presents a new application specific integrated circuit (ASIC) with advanced image correction architectures to provide a seamless, on-chip solution to enhance the performance of infrared imagers based on CMOS and MEMS technologies.

The ASIC consists of 5 main components, including a serial interface module, a controller unit, a configuration memory, an image memory, and a calibration memory. The communication between the ASIC and the external firmware is handled by serial interface module. The controller unit generates the necessary timing signals to generate the image and performs the image correction operations to the streaming image like automatic readout gain calibration, correlated double sampling (CDS), on-chip averaging, and dead pixel correction. The configuration memory holds 63x16 setting bits. The image memory has a size of 17x16x16 bits and is responsible for storing the final infrared image. The calibration memory has 16x20x7 bits to store the 7 bit gain settings for each pixel.

The ASIC is designed using the cutting-edge simulation, synthesis, and layout generation tools, and all coding is done in Verilog HDL. The design is manufactured in a 0.18 μm CMOS technology, and the fabricated chip is tested exhoustively to

verify all of the functions. The ASIC utilizes about 41 K logic cells, occupies 1.233 mm^2, and consumes only 2.30 mW, while operating from a 1.8 V supply voltage and running at a 5 MHz clock frequency. The automatic readout gain calibration reduces the standard deviation of the pixel signals by 39% while the correlated double sampling removes the readout offsets and reduces the standard deviation among the columns by 32%. The on-chip averaging successfully suppresses the high frequency noises and achieves 82% reduction on the signal noise, and the dead pixel correction properly eliminates the dead pixels. The results show that ASIC implementation of the image correction algorithms greatly enhance the image quality of the CMOS infrared imagers.

Keywords: ASIC, Non-Uniformity, CDS, Averaging, Dead Pixels, CMOS Infrared Imagers

ÖZ

CMOS KIZILÖTESİ GÖRÜNTÜLEME SENSÖRLERİ İÇİN GELİŞMİŞ GÖRÜNTÜ DÜZELTME MİMARİLERİNİ İÇEREN UYGULAMAYA ÖZEL TÜMDEVRE TASARIMI

Cilbir, Görkem
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Tayfun Akın

Haziran 2024, 110 sayfa

Bu tez, CMOS ve MEMS teknolojilerine dayalı kızılötesi görüntüleyicilerin performansını artırmaya yönelik kesintisiz, çip üzerinde çözümler sağlamak için ileri seviye görüntü düzeltme mimarilerileriyle desteklenmiş bir uygulamaya özel tüm devre (ASIC) sunmaktadır.

Tümdevre, seri arayüz modülü, kontrol birimi, ayarlama hafızası, görüntü hafızası ve kalibrasyon hafızası olmak üzere beş bileşenden oluşmaktadır. Tümdevre ve kullanıcı arasındaki iletişim seri arayüz tarafından gerçekleştirilmektedir. Kontrol birimi görüntü üretebilmek için gereken zamanlama sinyallerini üretmekte ve akan görüntü üzerinde otomatik kazanç kalibrasyonu, ilintili çifte örnekleme (CDS), çip üzerinde ortalama ve ölü piksel düzeltme gibi görüntü düzeltme operasyonlarını gerçekleştirmektedir. Ayarlama hafızası 63x16 ayar biti tutabilmektedir. Görüntü hafızası 17x16x16 bit boyutundadır, ve işlenmiş görüntüyü depolamakla yükümlüdür. Kalibrasyon hafızası 16x20x7 bitten olusur ve her bir piksel icin 7 bitlik kazanç ayarini tutar.

Sunulan tümdevre, ileri teknoloji benzetim, sentez ve serim araçları kullanılarak tasarlanmıştır, ve kodlama Verilog Donanım Tanımlama Dili'nde yapılmıştır. Tasarım, 0.18 µm CMOS teknolojisinde üretilmiştir, ve üretilen çip, tüm işlevlerinin doğrulanması amacıyla etraflıca test edilmiştir. Tümdevre, 1.8 V'luk besleme voltajı ve 5 MHz saat frekansında çalışmakta, yaklaşık 41 K mantık hücresi ile 1.233 mm^2 alan kaplamakta, ve 2.30 mW güç tüketmektedir. Otomatik kazanç kalibrasyonu pikseller arasındaki standart sapmayı %39 azaltmaktadır. İlintili çifte örnekleme, okuma devresinin getirdiği sapmayı çıkartmakta ve kolonlar arasındaki standart sapmayı %32 azaltmaktadır. Çip üzerinde ortalama, yüksek frekanslı gürültüleri başarılı şekilde baskılayarak sinyal gürültüsünü %82 azaltmaktadır. Ölü piksel düzeltme uygulaması da ölü pikselleri öngörüldüğü şekilde ortadan kaldırmaktadır. Sunulan tümdevre CMOS kızılötesi görüntüleyicilerinin görüntü kalitesini büyük oranda iyileştirmektedir.

Anahtar Kelimeler: ASIC, Düzensizlik Giderme, CDS, Çip Üzeri Ortalama, Ölü Pikseller, CMOS Kızılötesi Görüntüleyiciler

To my beloved family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Prof. Dr. Tayfun Akın for his supervision, advice, and valuable support throughout the research.

I would like to thank my beloved friend Hande Öztürk not only for all her ideas, guidance, and support during my thesis study, but also for her invaluable friendship. I am grateful to my dear friend Ferhat Taşdemir who always enlightens me with his incentive comments, suggestions, and support. I am grateful to Orhan Akar for sharing his in-depth knowledge and support throughout the study.

I want to thank my dear friends Volkan Coşkun and Nihan Çakır for their priceless ideas, comments, and friendships. I would like to express my appreciation to all Mikrosens Inc. staff who shares remarkable memories with me.

I also would like to thank Mikrosens Inc. for providing access to the IC design software and fabricating the ASIC implemented in this thesis.

Finally, I want to express my love and gratitude to my family for their endless encouragement and patience. Their love and support always have been my main motivation.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ALGORITHMS xx

CHAPTERS

1 INTRODUCTION 1

1.1 Infrared Imaging Systems 3

1.2 Image Artifacts 5

1.2.1 Pixel and Circuit Non-uniformities 6

1.2.2 Noise 8

1.2.3 Dead Pixels 9

1.3 Motivation of the Thesis 10

1.4 Research Objectives and Thesis Organization 11

2 SYSTEM OVERVIEW AND RESEARCH 13

2.1 Introduction 13

2.2 Automatic Readout Gain Calibration 13

2.2.1 Bias Voltage Calibration 14

2.2.2 Readout Gain Calibration 17

2.2.3 Comparison of the Results 19

2.3 Correlated Double Sampling 19

2.3.1 Analog CDS 20
2.3.2 Digital CDS 21
2.3.3 Proposed Method 23
2.4 On-Chip Averaging 23
2.5 Dead Pixel Correction 28
3 CIRCUIT DESIGN AND IMPLEMENTATION 39
3.1 Introduction 39
3.2 Top Level Architecture 39
3.3 Serial Interface 42
3.4 Controller Unit 51
3.4.1 Introduction 51
3.4.2 Image Generation 53
3.4.3 Automatic Readout Gain Calibration Implementation 57
3.4.4 Correlated Double Sampling Implementation 61
3.4.5 On-Chip Averaging Implementation 65
3.4.6 Dead Pixel Correction Implementation 68
3.4.7 Controller Unit Layout 72
3.5 Memory Blocks 74
3.5.1 Introduction 74
3.5.2 Configuration Memory 74
3.5.3 Image Memory 78
3.5.4 Calibration Memory 81
4 VERIFICATION AND TEST RESULTS 85
4.1 Introduction 85

4.2 Test Setup 85
4.3 Firmware Development 87
4.4 Verification of The Functions 88
4.4.1 Serial Interface and Memory Verifications 88
4.4.2 Automatic Readout Gain Calibration Verification 95
4.4.3 Correlated Double Sampling Verification 96
4.4.4 On-Chip Averaging Verification 98
4.4.5 Dead Pixel Correction Verification 100
5 CONCLUSION AND FUTURE WORK 103
REFERENCES 107

LIST OF TABLES

TABLES

Table 1.1 Parameter deviations in 0.18 µm CMOS technology. 6
Table 2.1. On-chip averaging coefficients. 25
Table 3.1. ASIC design specifications. 41
Table 3.2. Eminent features of the sub-blocks of the ASIC. 41
Table 3.3. Memory address decoder bits and corresponding memories. 43
Table 3.4. Pin functions of the SPI peripheral circuit. 44
Table 3.5. Pin functions of the I2C peripheral circuit. 48
Table 3.6. Final design specifications of the serial interface module. 50
Table 3.7. Final design specifications of the controller unit. 72
Table 3.8. Final design specifications of the configuration memory. 76
Table 3.9. Final design specifications of the image memory. 79
Table 3.10. Final design specifications of the calibration memory. 84
Table 4.1. Peak standard deviation values (16-bit ADC count) of the pixels along 100 frames before and after 30 stage on-chip averaging. 99

LIST OF FIGURES

FIGURES

Figure 1.1. Infrared bands of the electromagnetic spectrum [2]. 1
Figure 1.2 Infrared imaging system. 4
Figure 1.3. (a) Sample image with fixed pattern noise (FPN) (b) FPN removed. 7
Figure 1.4. Various noises added to the infrared signal. 8
Figure 1.5. SEM image of a CMOS infrared sensor pixels. Dead pixel is marked with a red "X" 9
Figure 2.1. Simulation results of the bias voltage calibration method: (a), (d), (g) and (j) show the progress of the printed image; (b), (e), (h) and (k) show the progress of pixels along FPA; and (c), (f), (i) and (l) show the progress of the pixel histograms. 16
Figure 2.2. Simulation results of the readout gain calibration method. (a), (d), (g) and (j) show the progress of the printed image, (b), (e), (h) and (k) show the progress of pixels along FPA, (c), (f), (i) and (l) show the progress of the pixel histograms. 18
Figure 2.3. Analog CDS implementation circuit [12]. 21
Figure 2.4. Block diagram of an off-chip digital CDS implementation circuit [13]. 22
Figure 2.5. Simulation results of ideal averaging vs the proposed method. Blue line shows the standard deviations of the results obtained by the ideal calculation, red line shows the standard deviations of the results obtained by the proposed method. 27
Figure 2.6. Possible dead pixel locations where the first correction scheme is applied. 29
Figure 2.7. Pixel designations used in the first scheme. 30
Figure 2.8. Possible dead pixel locations where the second correction scheme is applied. 31

Figure 2.9. Pixel designations for the top-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array. 32
Figure 2.10. Pixel designations for the right side group of the second scheme. Pixels with darker grey color are not part of the focal plane array. 32
Figure 2.11. Pixel designations for the bottom-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array. 33
Figure 2.12. Pixel designations for the left-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array. 34
Figure 2.13. Possible dead pixel locations where the third correction scheme is applied. 35
Figure 2.14. Pixel designations for the top-left corner of the third scheme. Pixels with darker grey color are not part of the focal plane array. 36
Figure 2.15. Pixel designations for the top-right corner of the third scheme. Pixels with darker grey color are not part of the focal plane array. 36
Figure 2.16. Pixel designations for the bottom-right corner of the third scheme. Pixels with darker grey color are not part of the focal plane array. 37
Figure 2.17. Pixel designations for the bottom-left corner of the third scheme. Pixels with darker grey color are not part of the focal plane array. 38
Figure 3.1. Top level architecture of the ASIC along with the other major blocks. 40
Figure 3.2. Top level layout of the ASIC. The total layout area of the circuit is 1233043 μm^2 in 0.18 µm CMOS technology. 40
Figure 3.3. Simulation results of an SPI data package for single word write mode. 44
Figure 3.4. Simulation results of SPI data packages for single word read mode. ... 45
Figure 3.5. Simulation results of video output mode of SPI. 46
Figure 3.6. Simulation results of the custom handshake mechanism. 46
Figure 3.7. Simulation results of an I2C data package for single word write mode. 48
Figure 3.8. Simulation results of an I2C data package for single word read mode. 49
Figure 3.9. Simulation results of video output mode of I2C. 50

Figure 3.10. Final layout view of the serial interface module. The total layout area of the circuit is 404.3 µm x 134.4 µm in 0.18 µm CMOS technology. 51
Figure 3.11. Flow diagram of the top state machine of the controller unit..... 52
Figure 3.12. Simulation results of the image generation. Rows are selected one by one. After all row reading is finished, infrared image for one frame is stored in the image memory and this is flagged to the external firmware by EOC bit..... 56
Figure 3.13. Simulation results of the image generation. Columns are selected one by one inside a row reading time. 57
Figure 3.14. General view of the simulation results of the automatic readout gain calibration. *cal_counter* indicates the current calibration stage..... 60
Figure 3.15. Detailed view of the simulation results of the automatic readout gain calibration. 61
Figure 3.16. General view of the simulation results of the correlated double sampling. *PC_READ_VINT* and *PC_READ_VRST* signals indicate integration state and reset states of the readout circuit. 64
Figure 3.17. Detailed view of the simulation results of the correlated double sampling. 65
Figure 3.18. Simulation results of the on-chip averaging. 68
Figure 3.19. Simulation results of the dead pixel correction for the bottom-right corner pixel. 71
Figure 3.20. Final layout view of the controller unit. The total layout area of the circuit is 354.5 µm x 504.6 µm in 0.18 µm CMOS technology. 73
Figure 3.21. Schematic view of a section of the configuration memory for one bit. 75
Figure 3.22. Final layout view of the configuration memory. The total layout area of the circuit is 288.6 µm x 570 µm in 0.18 µm CMOS technology. 77
Figure 3.23. Schematic view of a section of the image memory for one bit. 78
Figure 3.24. Final layout view of the image memory. The total layout area of the circuit is 379.7 µm x 1024.6 µm in 0.18 µm CMOS technology. 80
Figure 3.25. Schematic view of a section of the calibration memory for one bit.. 82

Figure 3.26. Final layout view of the 7-bit unit cell of the calibration memory. (a) RTL to GDSII flow implementation, (b) custom implementation. The total layout areas of the circuits are 124.3 µm x 17.1 µm for the first circuit, and 89.5 µm x 14.5 µm for the second circuit in 0.18 µm CMOS technology. 83
Figure 3.27. Final layout view of the calibration memory. The total layout area of the circuit is 1856.4 µm x 240.5 µm in 0.18 µm CMOS technology. 84
Figure 4.1. Test setup. 86
Figure 4.2. Verification results SPI single word write operation. Changing the content of the particular configuration memory address changes the states of the digital test outputs DT2 and DT1. 89
Figure 4.3. Verification results I2C single word write operation. Changing the content of the particular configuration memory address changes the states of the digital test outputs DT2 and DT1. 89
Figure 4.4. Verification results SPI single word read operation. Content of the configuration memory address 0x4033 is read back through the SPI as 0x01E4. .. 90
Figure 4.5. Verification results I2C single word read operation. Content of the configuration memory address 0x4033 is read back through the I2C as 0x01E4. .. 91
Figure 4.6. Verification results calibration memory. Content of the calibration memory address 0x838F is read back through the SPI as 0x36. 92
Figure 4.7. Verification results image memory. Content of the image memory address 0xC389 is read back through the SPI as 0x553E. 92
Figure 4.8. Verification results SPI video output mode. The SPI package in the figure carries an infrared video frame of 17x16x16-bit pixels. 93
Figure 4.9. Verification results I2C video output mode. The I2C package in the figure carries an infrared video frame of 17x16x16-bit pixels. 94
Figure 4.10. Verification results of the automatic readout gain calibration. DT1 is set to busy signal of the controller unit, and DT2 is set to vsync. 95
Figure 4.11. Verification result of the automatic readout gain calibration. (a) Histogram of the pixels before the automatic readout gain calibration. (b) Histogram of the pixels after the automatic readout gain calibration. 96

Figure 4.12. Verification results of the CDS operation. (a) DT2 is set to *Vint* signal of the controller unit. (b) DT2 is set to *Vrst* signal of the controller unit. 97
Figure 4.13. Verfication result of the correlated double sampling. (a) Histogram of the pixels before the correlated double sampling. (b) Histogram of the pixels after the correlated double sampling. On-chip offset is also added to the after CDS image to approximate the histogram limits to the before CDS image as well as to test the internal offset function. 97
Figure 4.14. Verification results of the on-chip averaging operation. DT1 is set to adc clock signal of the controller unit, and DT2 is set to *vsync* signal of the controller unit. 98
Figure 4.15. Pixel values along 100 frames (a) before and (b) after 30 stage on-chip averaging. Histograms of the standard deviations of the pixels along 100 frames (c) before and (d) after 30 stage on-chip averaging. 99
Figure 4.16. Verification result of the dead pixel correction. (a) Image before the dead pixel correction. (b) Image after the dead pixel correction. 100
Figure 4.17. Some pictures captured using the ASIC and accompanying 16x16 CMOS-IR imager. Upper two images show a human face. Lower left image shows fingers grasping a hot cup, and lower right image shows a hot cup and a cold cup together. 101

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1. Image generation. 55
Algorithm 2. Automatic readout gain calibration. 59
Algorithm 3. Correlated double sampling. 63
Algorithm 4. On-chip averaging. 67
Algorithm 5. Dead pixel correction. 70

CHAPTER 1

INTRODUCTION

Light is the main medium of information to visual senses of mankind for ages. However, the partition of the light spectrum is not well known until Sir Isaac Newton and the famous prism experiment, which showed that the light spectrum lays between red and violet visible lights. Figure 1.1 illustrates the infrared bands of the electromagnetic spectrum. Further observations conducted by Wiliam Herschel proved that there is more in the spectrum beyond our vision and infrared region is surely one of them. For a long time since its first discovery by Herschel, the infrared light and its applications were in crawling. This situation changed in the mid-20th century with the discovery of the first infrared sensors [1]. Since then, infrared imagers have been evolved much and involved in daily life increasingly.

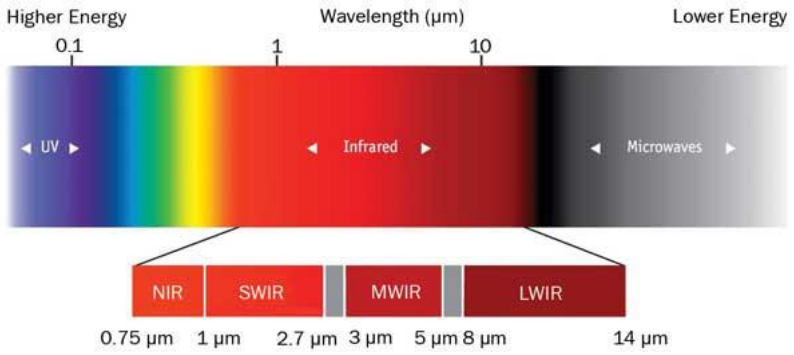


Figure 1.1. Infrared bands of the electromagnetic spectrum [2].

The two main types of infrared imagers are photonic infrared imagers and thermal infrared imagers. Photonic infrared imagers are based on semiconductor materials which are manufactured to detect a specific infrared band of the spectrum. Generally,

they require additional cooling systems, and they are used in high end cameras since their manufacturing cost is expensive. Thermal infrared imagers, on the other hand, utilize basic microstructures and take advantage of the change of the electrical properties of these structures as they are exposed to the infrared radiation. CMOS infrared (CIR) imagers can be considered as a subbranch of the thermal imagers. They are manufactured in a classical CMOS manufacturing flow and pixels are formed by basic MEMS steps afterwards. They do not require sophisticated compounds and are easy to manufacture, therefore they are relatively low cost compared to the other detector types.

Infrared imagers are used in diverse areas like astronomy, military, medical, automotive, environment observation, smart buildings etc. and each application has its own requirements like infrared band, sensitivity, resolution, frames per second. Infrared imagers have different readout circuit topologies depending on the detector type and the application area. These readout circuits are designed to get the optimum performance from the sensor for the targeted application.

Various artifacts like assorted noises and dead pixels may be added to the final infrared image from detector signal through the readout circuit. Some readout circuits have special structures or accompanying ASICs to remove those artifacts and correct the image, whereas others leave the image correction job outside of the imager. As technology advances, such correction circuits cover less layout area and consume less power and hence make it possible to implement such image correction architectures on-chip. An infrared imager, external electronics like regulators, ADCs and DACs, external processing units like FPGAs, MCUs, frame grabbers and an image displaying device constitute the basic elements of a complete infrared imaging system.

This thesis compiles the research work on image artifacts observed in the CMOS infrared imagers and presents an ASIC with on-chip circuit architectures to correct those artifacts and hence to obtain better image quality. The ASIC is used for controlling the CMOS infrared imager as well as implementing the correction

algorithms to the streaming image, while eliminating the need for extra outside components for this purpose.

This chapter gives basic information about the infrared imaging systems, image artifacts found in CMOS infrared imagers, motivation of the study as well as the thesis organization. Section 1.1. introduces a complete infrared imaging system in general. Section 1.2. defines the unwanted image artifacts and their possible sources. Section 1.3. states the goals and motivation of this thesis. Finally, section 1.4. Lists the research objectives and thesis organization.

1.1 Infrared Imaging Systems

Detector structures in an infrared imager generates a signal due to incoming infrared radiation. These signals are collected and sorted by the readout circuit of the imager and streamed out. However, these imagers cannot construct meaningful infrared images by themselves. They need to be driven by an external supervising device like field programmable gate array (FPGA), microcontroller unit (MCU), frame grabber etc. In some cases, the readout circuit of the imager does not include an on-chip analog to digital converter (ADC) and gives analog output. In such cases, analog output of the imager is sampled and converted to digital domain by an external ADC. In addition, most of the infrared imagers require a proper external biasing. Discrete components like regulators, capacitors and filters carry out the biasing. Finally, a displaying device, most likely a computer, displays the final infrared image and applies further processing, if necessary. All the external hardware along with the infrared imager itself constitutes a complete infrared imaging system. Figure 1.2 demonstrates a sample infrared imaging system and its sub-blocks.

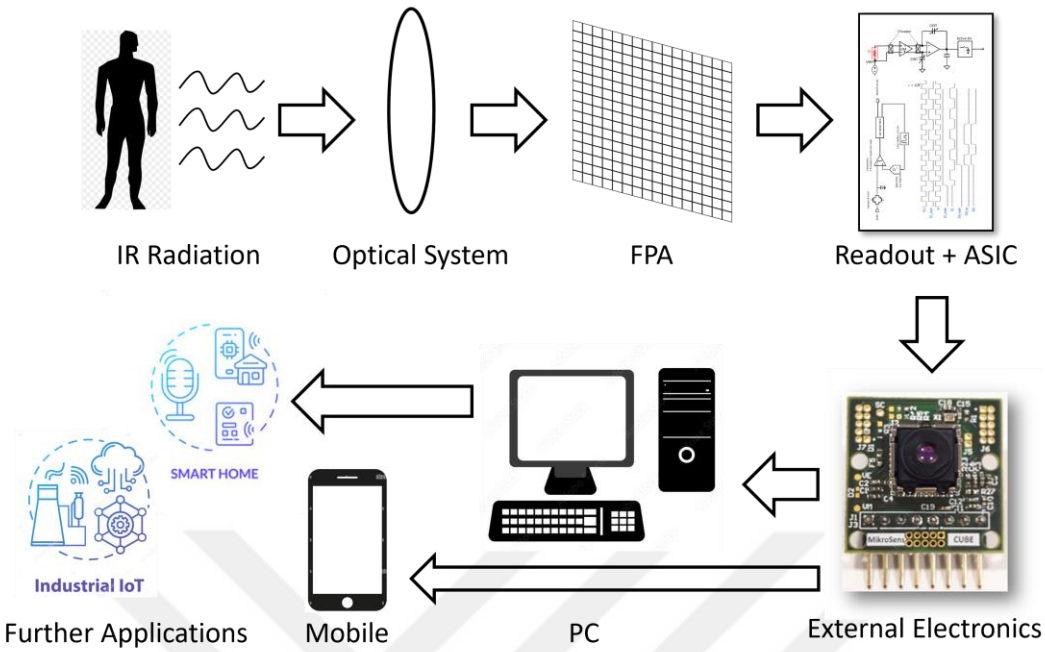


Figure 1.2 Infrared imaging system.

Infrared imaging systems are designed to meet the requirements of a specific application. These requirements determine the important parameters of an infrared imaging system. These parameters include sensitivity, array size, frame rate, power consumption, chip size and data transmission protocols, and most of them are directly related to the imager design.

Sensitivity is one of the most important parameters which designates the overall performance of the infrared imaging system. The total illuminated area of a detector and material properties affect the detector sensitivity and therefore the infrared signal level. The readout circuit architecture and external components are mostly shaped according to this signal level.

The array size is another prominent parameter of an infrared imaging system since it forms the final image aspect ratio. Low format arrays meet the requirements for some applications like handheld thermometers and various machine vision systems.

However, higher array sizes are required for some other applications like space observation and military.

Frame rate is the streaming image frames per second. More frames mean more information input for an application. Higher frame rates give smoother video output. In addition, with the higher frame rates, frame averaging and hence suppression of noise can be achieved.

The power consumption becomes more important as new technologies like wearables and internet of things emerged. A low power design is a must for handheld devices and battery powered systems. In addition, an excessive power consumption heats up the imager, and therefore, distorts the infrared signal.

Communications of the sub-blocks with each other in an infrared imaging system is carried out by common protocols like SPI, I2C, UART and USB. Most of the external driving devices support at least one of these protocols. Therefore, an on-chip serial interface which implements the contemporary communication protocols makes it easy to integrate the imager to the system and lowers the system cost.

1.2 Image Artifacts

A good system level design may prevent most of the possible problems of an infrared imager. Nevertheless, the final image may incorporate some undesirable artifacts inevitably. Presence of these artifacts yields unpleasant results in the final image. Major image artifacts include pixel and circuit non-uniformities, various noises, and dead pixels. Some of them are device related noises, whereas some are system level artifacts. These artifacts should be corrected to get the optimum performance from an infrared imaging system. There are particular mitigation techniques to eliminate those artifacts. To develop proper elimination algorithms tailored to a particular infrared imager, those image artifacts should be examined first. Next sections will introduce those artifacts.

1.2.1 Pixel and Circuit Non-uniformities

State of the art CMOS infrared detectors are produced using CMOS and MEMS processes. It is not practically possible to completely eliminate the errors caused by the mismatches during the CMOS fabrication. Source of CMOS mismatches can be lithographic variations, line-edge roughness, random dopant fluctuations, layout-dependent stress variations, rapid thermal annealing (RTA) temperature induced variations, well-proximity effects (WPE), deposition and growth processes, and chemical mechanical polishing (CMP) [3]. In general, the percentage deviations in physical dimensions, threshold and gain factor are modeled as pelgrom coefficients [4], [5]. Table 1.1 gives a summary of some parameter deviations reported in a 0.18um CMOS technology. Furthermore, MEMS processes variations bring geometry variations in the focal plane array (FPN). These non-uniform pixels respond the incoming infrared light differently. Hence generated pixel signals differ even though they are exposed to the same infrared light.

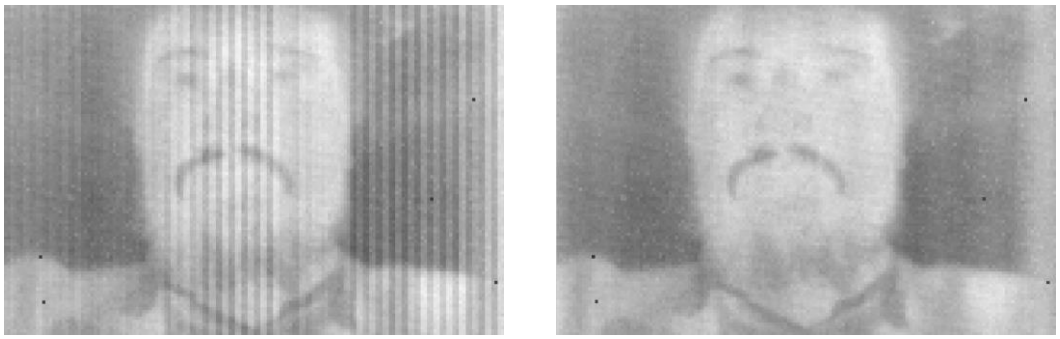
Table 1.1 Parameter deviations in 0.18 μm CMOS technology.

Pelgrom coefficient	**Value**
pmos gain factor mismatch	0.78 %μm
pmos threshold voltage mismatch	6.72 mVμm
nmos gain factor mismatch	1.72 %μm
nmos threshold voltage mismatch	13.8 mVμm
resistor mismatch	1.65 %μm
capacitor mismatch	0.16 %μm

Effects of these mismatches cause non-uniform pixel responsivities at the FPA and gain differences in the readout circuit. Combination of all these effects create a non-uniformity at the signal slopes against varying infrared light intensity. In the final

image this shows up as faint and bright pixels as scene changes. Impacts of this gain variations can be eliminated by adjusting the readout gain accordingly. Therefore, the readout circuit should be calibrated regarding the pixel signal variations and CMOS mismatches.

Another major artifact found in the infrared imagers is offset. When exposed to the infrared light, pixels in the imager generate a very small amount of analog signal. The readout circuit amplifies this signal to a workable level. This amplification typically adds an offset to the pixel signal. This offset differs between the readout circuit channels due to CMOS mismatches and emerges as column offset differences in the final image. In addition, MEMS process variations on the focal plane array bring extra spatial variations on the pixel signals, if the pixels are current biased type. In this case, these spatial variations appear in the whole pixel array and reflected to the final image as fixed pattern noise (FPN). In order to construct meaningful image, the offset part should be removed from the amplified pixel signal. This can be achieved by removing the reset state signal of the readout from pixel reading state signal. Figure 1.3 shows sample images with FPN and FPN removed.



(a) (b)

Figure 1.3. (a) Sample image with fixed pattern noise (FPN) (b) FPN removed.

1.2.2 Noise

Infrared imagers employ various noise types, changing in intensity, spatial and temporal features depending on the detector and readout circuit. These noises become superimposed on the infrared signal and cause image degradation since the final image exhibits a signal-to-noise ratio lower than 1 [6]. Figure 1.4 depicts the noise sources added to the infrared signal. The 4kTR noise and kTC noises are introduced by the pixel array and the readout circuit. These noises appear as white noise in the final image. There are certain image processing techniques to filter out these unwanted noises and to improve the image quality. Stochastic noises can be eliminated either by modulator-demodulator structures and analog filter circuits or by digital image processing algorithms in the system level.

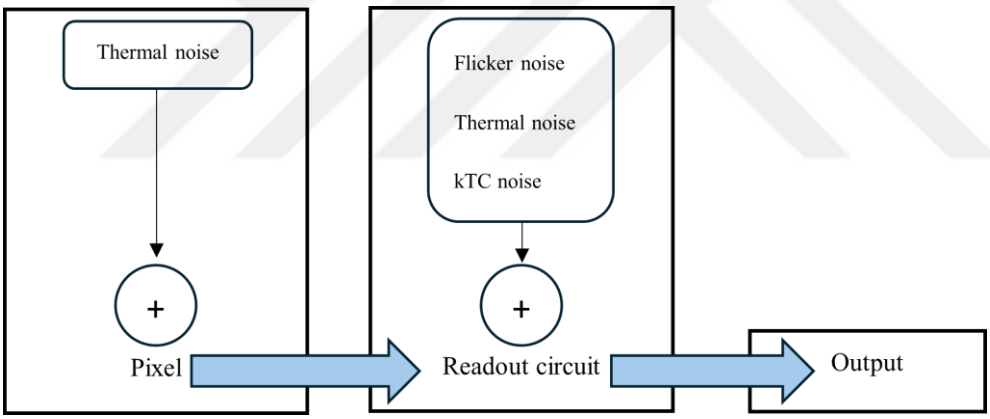


Figure 1.4. Various noises added to the infrared signal.

1.2.3 Dead Pixels

During the manufacturing process of the infrared detectors some defects or deformations may occur. Figure 1.5 shows scanning electron microscope images of a deformed pixel. These defects lead to deform the physical structure of the detector and eventually degrade the overall detector performance. Some of the deformed detectors are abnormally responsive to the incoming infrared light whereas some of them are not responsive at all. Output of such deformed detectors appears as dead pixels in the final image. Dead pixels severely degrade the overall image quality therefore they are unwelcomed in most cases.

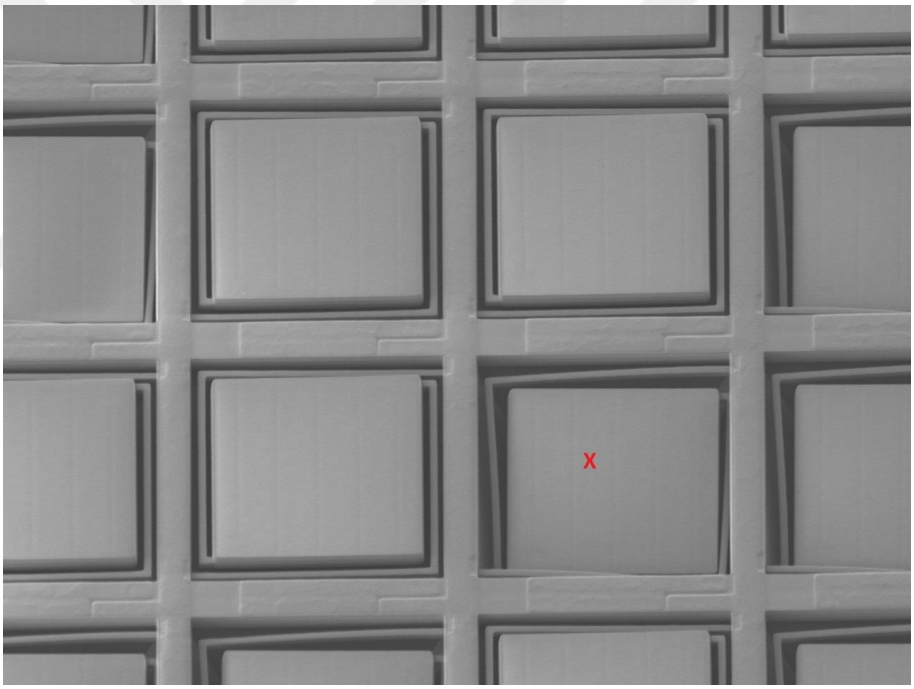


Figure 1.5. SEM image of a CMOS infrared sensor pixels. Dead pixel is marked with a red “X”.

1.3 Motivation of the Thesis

The motivation of this thesis is to design and implement an ASIC that employs advanced image correction architectures for CMOS infrared imagers, and it aims to eliminate common undesirable infrared image artifacts with on-chip implementations of novel algorithms therefore to reduce the overall power consumption and system cost. Image correction methods usually implemented in off-chip devices. This brings extra power consumption and system cost as well as extra development effort. In addition, off-chip components introduce additional noise yielding degraded signal integrity of the imaging system. On the other hand, on-chip implementations like the ASIC proposed in this thesis, are tailored to meet the requirements of CMOS infrared imagers as they give an integrated solution to image correction problem and remove the need for extra components. This dramatically reduces the power consumption and the system cost. Furthermore, with on-chip solutions, better signal integrity is achieved since the additional noise from external components are not in question.

This thesis presents an ASIC with advanced on-chip image correction architectures to drive the sensor and to remove the image artifacts. Novel image correction algorithms are developed and implemented on-chip as well as auxiliary memory blocks and a flexible serial interface. The proposed ASIC design asserts an all-in-one solution to the image correction problem, reducing the overall power consumption and system cost, while achieving a better signal integrity.

1.4 Research Objectives and Thesis Organization

The main objective of this thesis is to develop an ASIC to drive an infrared imager and to implement image correction algorithms on the streaming image. The proposed design should achieve low power consumption and should occupy low layout area. To achieve 30 frames per second throughput while applying the corrections, it should work at 5 MHz clock speed and should apply the correction operations to 16-bit pixel data. In addition, the proposed design should accommodate various memories to configure and hold the image, and a serial interface to receive the algorithm parameters and to stream the image out. The main objectives of the study are given below.

1. To design a serial interface which supports both SPI and I2C peripherals with video output mode.
2. To design a controller unit which drives the pixel array and analog blocks and employs the image correction architectures such as readout gain calibration, digital correlated double sampling, frame averaging and dead pixel correction.
3. To design a configuration memory to hold the settings of the ASIC and analog blocks.
4. To design an image memory to store the image and to serve as a cash memory to the image correction algorithms.
5. To design a calibration memory to hold the values of the programmable capacitor array inside the readout circuit.

Section 2 investigates the image correction algorithms in literature and introduces the developed techniques to implement these algorithms on-chip. Section 3 gives top level architecture of the design and implementation details and features of the sub-blocks of the ASIC. Section 4 demonstrates the verification and test results of the manufactured chip. Section 5 summarizes the work done and concludes the thesis and states the future work to further improve the performance of the chip.

CHAPTER 2

SYSTEM OVERVIEW AND RESEARCH

2.1 Introduction

This chapter focuses on analyses of the image correction methods in literature, compiles the research work for development of the image correction algorithms, and based on the research outputs, proposes the best solutions to implement these algorithms on-chip. Section 2.2 introduces automatic readout gain calibration, while Section 2.3 discusses correlated double sampling. Section 2.4 describes on-chip averaging, and finally, Section 2.5 demonstrates dead pixel correction algorithm.

2.2 Automatic Readout Gain Calibration

One of the outcomes of pixel and circuit non-uniformities is the gain differences among the amplified pixel signal levels. Effects of these non-uniformities should be eliminated prior to analog to digital conversion and further processing of the pixel signals.

Non-uniformities can be eliminated by smart correction methods. As an example, microbolometer type IR detectors are suspended, bridge like resistive structures. The incoming IR radiation heats up the pixel and IR signal is generated when the detector is biased. In addition to the infrared radiation, the heat that occurs due to the current passing through the microbolometer pixel is also sensed and this phenomenon is called as self-heating effect. This unwanted part of the signal can be subtracted by a proper reference detector. However, the non-uniformity in the detector or reference give limited subtraction and therefore remaining signal may saturate the sensor output signal.

One non-uniformity correction application is to program the reference detector [7]. Other non-uniformity correction application is to program the detector bias by programming a pixel biasing DAC [8]. Another non-uniformity correction application in literature is to program the bias voltage of the FPA [9].

A new readout architecture for the next generation infrared detectors is developed by Mikrosens. This readout architecture employs a pixel voltage bias generation and a capacitor array in the gain stage and these blocks are designed by analog design team of Mikrosens. In this thesis, a novel approach is presented to correct the non-uniformities in the readout architecture introduced by Mikrosens. This smart methodology utilizes the programming and auto calibration of the capacitor array in the column readout channel to control the gain. Considering the accuracy, performance and the area impacts, 7-bits resolution is decided for the calibration implementation. In order to estimate the outputs and proof the proposed calibration approach, a simulation model is created in MATLAB environment. In addition, a modified version of the bias voltage calibration method is also modelled and outputs of these two methods are compared. Further sections will explain these two simulation models in detail.

2.2.1 Bias Voltage Calibration

The first simulated approach is the bias voltage calibration method. Bias voltage calibration method aims to converge the pixel output to a predefined value by changing the pixel bias voltage. In this method, the pixel bias voltage is controlled by a dedicated voltage DAC. A simulation set up is created utilizing the bias voltage calibration method. In the model, a binary search algorithm is developed to automatize the calibration. Modelled ADC output for a pixel including the rest of the readout structure is defined by,

$$ADC\ Output\ Code(16\ bit) = \left[2^{16}.\left[V_{bias} + \left(\left(G_m.V_{pix}\right).\left(\frac{T_{int}}{C_{int}}\right)\right)\right]\right]/3.3 \quad (2.2.1)$$

where V_{bias} is the DAC output to be calibrated, G_m is the 100 µA/V transconductance constant, V_{pix} constant is the signals generated by the pixels due to infrared light. Dynamic range of V_{pix} is modelled to be in the range of 30 µV to 90 µV randomly. T_{int} has the constant value of 1 ms representing the integration time and C_{int} is the integration capacitance of the readout which is a constant value of 15.36 nF. In the model, the converge value for the binary search is set to be 33845 ADC counts. Binary search algorithm progresses bit by bit beginning from the MSB and completes the calibration in 7 steps. Initially the corresponding bit is set to 1 at the beginning phase of each step. Then the sensor output is calculated using the formula given in equation (2.2.1), and the calculated value is compared to the given converge value. Depending on the comparison, either the corresponding bit is kept as 1 or it is changed to 0. The binary search algorithm leaves the current bit as 1 if the output is less than the converge value and clears the current bit if the output is greater than the converge value. This flow will be repeated until the end of the 7th bit decision.

The DAC error coefficient is introduced as a non-ideality factor and DNL is set to be maximum of 1 LSB. The non-idealities in the rest of the readout circuit are not included in the analysis since these readout parts are common in both compared methods, and they will have less impact on the sensor output.

Results in Figure 2.1 demonstrates the improvement in the image with the bias voltage calibration method is applied. As shown in the figure, the standard deviation of the resulting array output is decreased from 2271 to 44. In addition, the mean of the resulting array output is 11 ADC counts close to the comparison value.

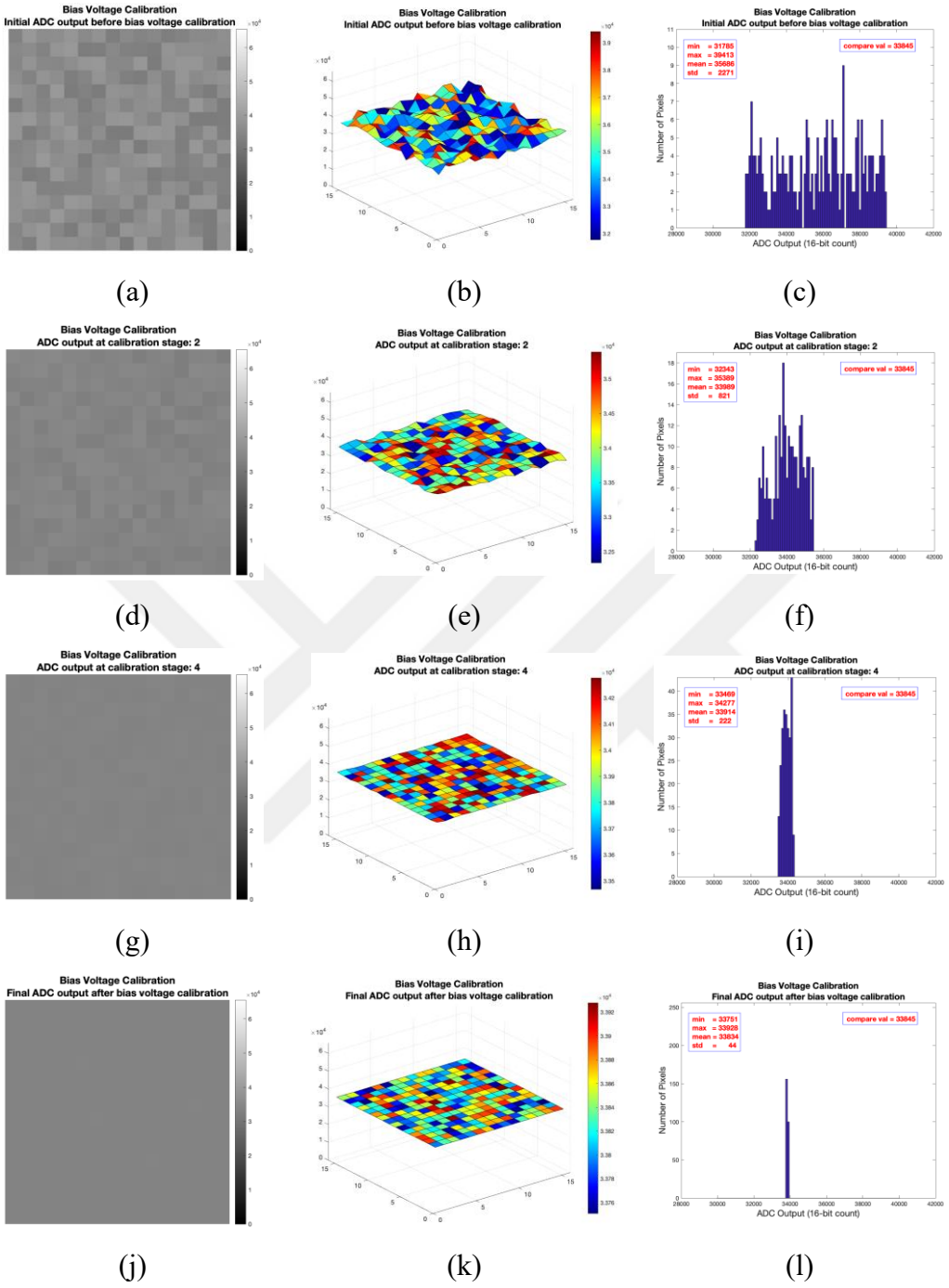


Figure 2.1. Simulation results of the bias voltage calibration method: (a), (d), (g) and (j) show the progress of the printed image; (b), (e), (h) and (k) show the progress of pixels along FPA; and (c), (f), (i) and (l) show the progress of the pixel histograms.

2.2.2 Readout Gain Calibration

The second simulated approach is the readout gain calibration method. The readout gain calibration method aims to converge the pixel output to a predefined value by changing the gain of the readout circuit. The second simulation setup is similar to the first one. The main difference from the first method is the calibration implementation. In the first method, value of C_{int} is kept as constant, and V_{bias} is calibrated. However, in the second method, V_{bias} is kept as constant, and C_{int} is calibrated. The V_{bias} value is set to 1.4 V. G_m has the same value as in the first simulation. The same randomly generated V_{pix} constant is used as in the first simulation. T_{int} is the constant value of 1 ms as in the first medhod representing the integration time, but this time C_{int}, the integration capacitance is the variable whose value is to be calibrated. The binary search algorithm for this method is slightly different from the first one. Since the gain is decreased as the C_{int} increases, binary search algorithm leaves the current bit as 1 if the output is greater than the converge value and clears the current bit if the output is less than the converge value. Finally, a voltage-DAC is not introduced in this method assuming as all pixels will be biased with the same bias voltage.

Results in Figure 2.2 demonstrates the improvement in the image with gain calibration. The standard deviation decreased from 2271 to 30 which is better than the first simulation results. However, the mean of the final pixel array is 41 ADC counts close to the comparison value which is worse than the first method.

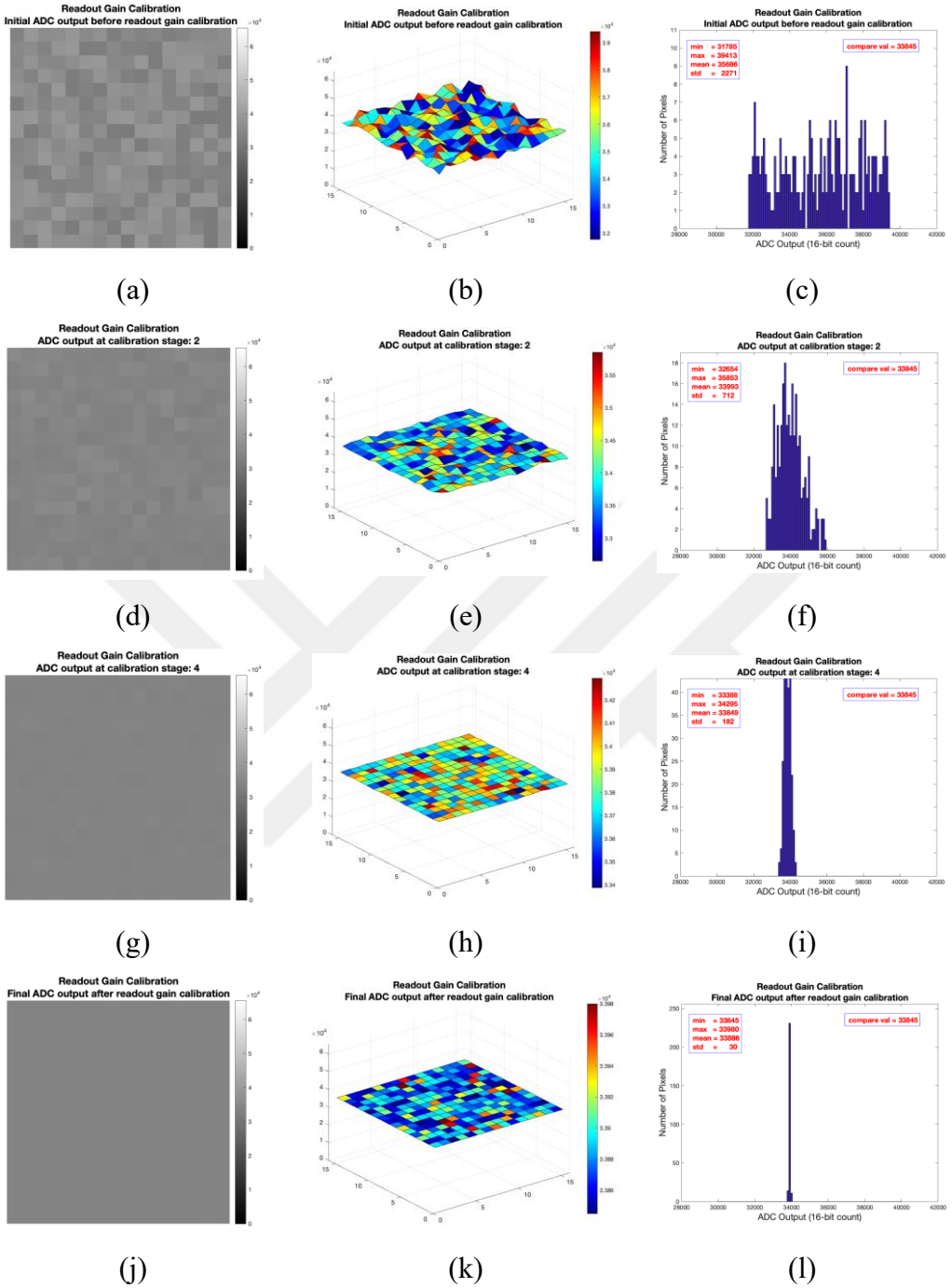


Figure 2.2. Simulation results of the readout gain calibration method. (a), (d), (g) and (j) show the progress of the printed image, (b), (e), (h) and (k) show the progress of pixels along FPA, (c), (f), (i) and (l) show the progress of the pixel histograms.

2.2.3 Comparison of the Results

It can be seen in the results that the improvement of standard deviation is better in the second approach, whereas the mean is closer in the first one. Since the aim of the proposed ASIC in this thesis is to correct the image artifacts of CMOS infrared imagers, improvement on the standard deviation is preferable compared to improvement on the mean. Besides, the non-linearities brought by DAC distorts the output signal. Furthermore, the DAC may also require high and low voltage limiting bias circuits which brings extra non idealities and extra design challenge. Finally, the bias voltage calibration method does not suggest a control over the readout gain mismatches. On the other hand, the second approach only requires the layout area of the extra capacitor arrays and the switches. Although the results of both approaches seem good enough to choose as the calibration method, drawbacks of the bias voltage calibration method mentioned above makes the gain calibration method is the preferred one to implement on the real hardware. In addition, preliminary work results show that a 7-bit implementation is enough for proper calibration of the readout circuit.

2.3 Correlated Double Sampling

One of the major problems found in infrared imagers is the fixed pattern noise (FPN). Infrared signal generated from a pixel is passed through readout circuit, and the non-idealities like noise, mismatch and non-linearities in the circuit add some unwanted offset to the sensor output signal. This unwanted offset which appears in the final image called as fixed pattern noise (FPN). The magnitude of the FPN signal can be hundred times greater than the infrared signal, so it may limit the imager dynamic range [10].

The basic way to remove the FPN in the final image requires storing at least two image frames, one when there is dark condition (no incoming light) and one there is

light. The difference of these two frames will result in an FPN-free sensor output. However, storing at least two image frame requires large memory [11].

The traditional solution to FPN problem is correlated double sampling (CDS) method. In the CDS method, the output of the readout circuit is sampled two times, but this happens in a row reading duration. One sample is read when the readout circuit is in reset state, and other one is sampled when the readout circuit is connected to the pixel. The sampled values are subtracted in further stages throughout the imager system. Depending on the implementation, CDS methods can be grouped into two main categories, analog CDS and digital CDS. Further sections will explain analog and digital CDS techniques.

2.3.1 Analog CDS

A common implementation of the CDS method is analog CDS, in which the reset and offset value of the readout circuit is subtracted from the pixel value through analog circuits like differential gain stages, differential input ADCs etc. An adopted version of the analog CDS implementation circuit proposed by [12] is demonstrated in Figure 2.3. In this implementation, there are two circuit stages, a capacitor transimpedance amplifier (CTIA) stage and a CDS stage. Whole operation can be summarized into three phases, CTIA reset phase, CTIA integration phase and CDS phase. In CTIA reset phase, CTIA circuit is kept in reset state and reset and offset voltage (*V1*) is accumulated on capacitor *C1*. Then in CTIA integration phase, pixel signal is integrated and added to the offset voltage (*V2*). Finally, in CDS phase, reset and offset voltages are cancelled out leaving only pixel signal at the output. Final output voltage is defined as:

$$-C_1V_1 = -C_1V_2 - C_2V_{out} \tag{2.3.1}$$

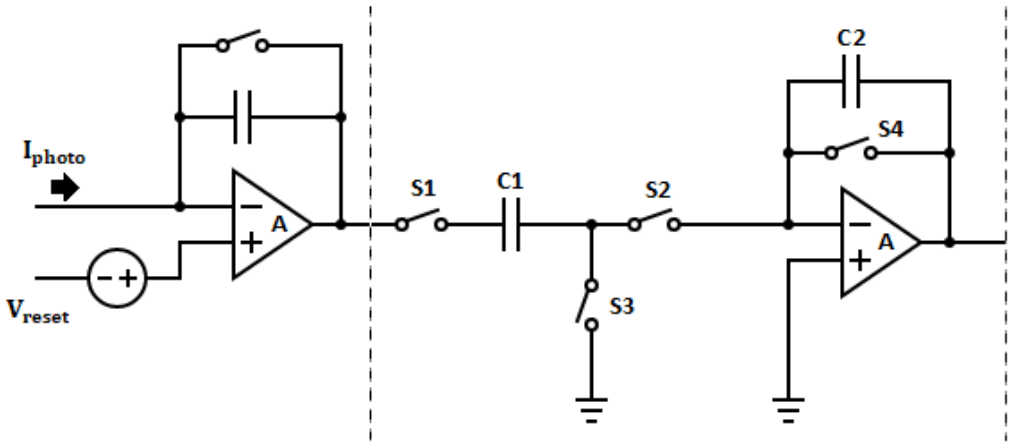


Figure 2.3. Analog CDS implementation circuit [12].

2.3.2 Digital CDS

Another implementation of the CDS utilizes digital circuits like processing units and memory blocks. Generally, in digital CDS methods, outputs of the readout circuit for reset state and pixel reading state stored in different memories and subtracted in processing units. Some of the CDS techniques are implemented off-chip. One example of off-chip implementation is introduced by [13]. Figure 2.4 illustrates the block diagram of this off-chip CDS implementation. In this implementation, reset signal and pixel signal is read by a sigma delta ADC and send to the FPGA outside of the chip. First, reset signals are read and stored on the FPGA. Then pixel signals are sent to the FPGA and subtraction takes place in this device.

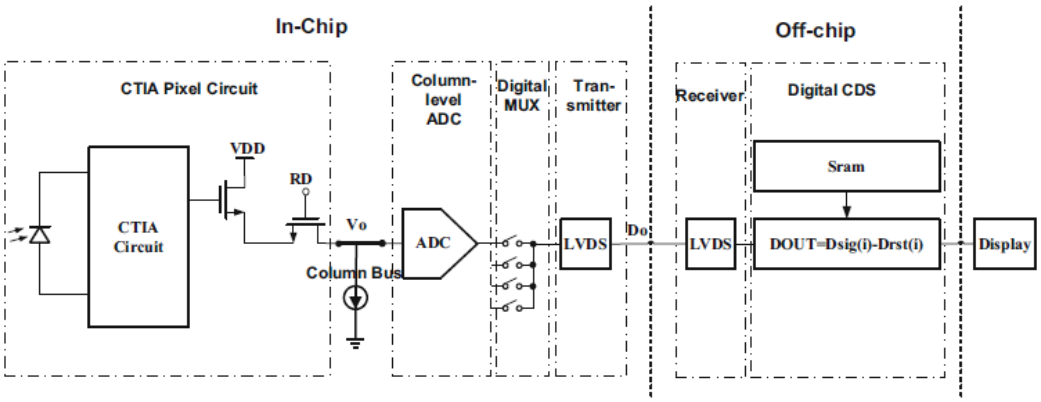


Figure 2.4. Block diagram of an off-chip digital CDS implementation circuit [13].

As pixel array sizes increases in CMOS imagers, it is getting increasingly difficult to convert all pixels with one ADC on-chip. Nowadays most of the CMOS infrared imagers have a single slope ADCs per column readout channel which are operating in parallel [14]. However, for low FPA sizes single on-chip ADC is enough to convert all pixels to digital without limiting the FPS. Presence of on-chip ADCs enable the on-chip implementations of digital CDS techniques.

In one of the early on-chip implementations of the digital CDS the reset signal and pixel signal is stored in latch type memories after single slope ADC conversion, and the reset signal is digitally subtracted from the pixel signal [15]. Another implementation improves the imager speed utilizing on chip CDS along with a differential difference amplifier (DDA) as comparator [16]. Here pixel signal and reset signals are accumulated on separate capacitors. At the same time a 10-bit counter drives a DAC, and the difference of voltage levels on capacitors is compared with the ramp signal generated by the DAC. The counter stops when the comparator output is asserted. Hence the last count value of the counter represents the digital values of the signals. Since this conversion is done one by one switching of the signal's, reset signal is removed in the final output count.

2.3.3 Proposed Method

The implementation of the CDS method in the proposed ASIC is decided to be digital and on-chip, comparing both analog and digital CDS methods and evaluating the on-chip and off-chip scenarios. Accompanying readout circuit to ASIC has a 16-bit SAR ADC and an image memory to store a frame of infrared image. Presence of these sub blocks gives flexibility to do the subtraction in the controller unit, digitally. The flow of the CDS implementation is given below:

1. The integration values of each pixel in a row are stored in the image memory.
2. The reset values are read from the readout circuit.
3. Just prior to storing the reset values in the image memory, integration values are fetched back to the controller unit from the image memory.
4. Digital subtraction operation is done by subtracting the reset signal from the integration signal.
5. The integration values in the internal memory are overwritten with the result of the subtraction operation.

Only few extra clock cycles are required for the CDS operation since the data flow between the controller unit and image memory is parallel. In addition, the non-linearities introduced by the ADC are also eliminated since the subtraction operation is done on digital data,

2.4 On-Chip Averaging

All CMOS infrared imagers inevitably suffer from intrinsic noise sources. Although they cannot be eliminated completely, some stochastic noises can be suppressed by particular techniques. Frame averaging is one of the common techniques to suppress noises like 4kTR noise and kTC noise and it helps to improve image quality obtained from a CMOS infrared imager. Since all these noise sources are random in nature, they can be eliminated by digital low pass noise filters like averaging of the ADC outputs at system level [17], [18].

In general, frame averaging is implemented on off-chip devices like FPGAs, MCUs and DSPs. However, this brings extra cost to overall system and may reduce the signal integrity. On the other hand, on-chip averaging solutions reduces the overall system cost since there is no need for extra processing devices in this case. Furthermore, on-chip averaging also helps to improve the signal integrity by offering a more compact data path within the chip.

It should be noted that implementing the on-chip averaging brings its own troubles. Traditional approaches for averaging include storing all incoming image frames and applying some arithmetic operations like summing and division. This classical approach requires considerable memory area to hold each incoming frame and relatively more clock cycles and high logic utilization to perform the arithmetic operations. The situation gets even worse as the averaging frame number increases. Both extra memory area and extra clock cycles directly affect the total power drawn from the chip and makes it harder to integrate it with the battery powered systems. On the contrary, there are some recursive averaging methods in the literature which can be implemented on-chip without requiring large memories, extra clock cycles and high logic utilization [19], [20]. A practical approach to implement the on-chip averaging which does not require these excessive sources is presented in this thesis work. Up to 30 frame averaging is implemented in the suggested method since 30 fps is a standard frame rate for infrared imagers.

The averaging method requires the determination of the two predefined coefficient values *C1* and *C2* for each averaging stage. Coefficients are defined in a way that they approximate the result to the ideal averaging. Table 2.1 shows the coefficients for each averaging stage. In the table, *C1* is the coefficient for the previous averaging result and *C2* is the coefficient for the new pixel value from the next frame. $n = 0$ means 1 frame averaging is applied which is in fact the same state as no averaging is applied, and $n = 4$ means 5 frame averaging is applied indicating 5 averaging stages. Same coefficients are used in some of the higher averaging numbers in order to utilize fewer resources in the proposed ASIC.

Table 2.1. On-chip averaging coefficients.

n (average count - 1)	C1	C2
0	0	32
1	16	16
2	21	11
3	24	8
4	26	6
5,6	27	5
7,8	28	4
9 to 13	29	3
14 to 20	30	2
21 to 29	31	1

A generalized equation for the suggested method is,

$$P_{(n)} = \frac{\left(C1 \,.\, P_{(n-1)} + C2 \,.\, P_{(ADC)}\right)}{32} \tag{2.4.1}$$

where,

n = average count – 1,

$P_{(n)}$ = calculated average value of the current stage,

$P_{(n-1)}$ = calculated average value of the previous stage,

$P_{(ADC)}$ = new pixel value from ADC

Averaging for n frame begins with the first averaging stage. Since there is no old averaging result for the first stage, old averaging result in the equation is regarded as 0 for this stage. For each new averaging stage, the new pixel value is multiplied with corresponding *C2* from the table, and old averaging result is multiplied with corresponding *C1* from the table. Then two values are summed up. Finally, this weighted sum is divided by 32 yielding the averaging result for that stage. The result

of an averaging stage is used as input for the calculation of the result of the next stage. The iteration is done after the calculation for the nth stage is completed. This procedure is applied every pixel in a frame.

An example calculation for 3 frame averaging for a pixel and calculation for ideal case for comparison are given below. Assuming 16-bit ADC output values for the same pixel for 3 consecutive frames are,

$P_{(ADC1)}$ = 23491, $P_{(ADC2)}$ = 23666, and $P_{(ADC3)}$ = 23857 and using the equation

$$P_{(n)} = \left(C1\,.\,P_{(n-1)} + C2\,.\,P_{(ADC)}\right)/32,$$

@ 1. averaging stage (1. frame arrived)

$$P_{(0)} = \frac{\left(0 + 32\,.\,P_{(ADC1)}\right)}{32} = 23491 \qquad (2.4.2)$$

@ 2. averaging stage (2. frame arrived)

$$P_{(1)} = \frac{\left(16\,.\,P_{(0)} + 16\,.\,P_{(ADC2)}\right)}{32} = 23578 \qquad (2.4.3)$$

@ 3. averaging stage (3. frame arrived)

$$P_{(2)} = \frac{\left(21\,.\,P_{(1)} + 11\,.\,P_{(ADC3)}\right)}{32} = 23673 \qquad (2.4.4)$$

whereas in the ideal case,

$$P_{(2)} = \frac{(23491 + 23666 + 23857)}{3} = 23671.33 \qquad (2.4.5)$$

The results show that the difference between the proposed method output and ideal averaging output is 1.67 ADC counts. This difference is negligible considering 16-bit ADC is used as input. A more general comparison of the proposed method with the ideal averaging case for all averaging stages can be seen in Figure 2.5. Comparison is made with randomly generated 16-bit numbers to mimic the ADC output. As seen in the figure, proposed method outputs mostly overlap with the ideal case outputs. However, as number of averaging count increases, the error also increases. This is mainly due to the coefficient limitation in higher averaging numbers and rounding errors.

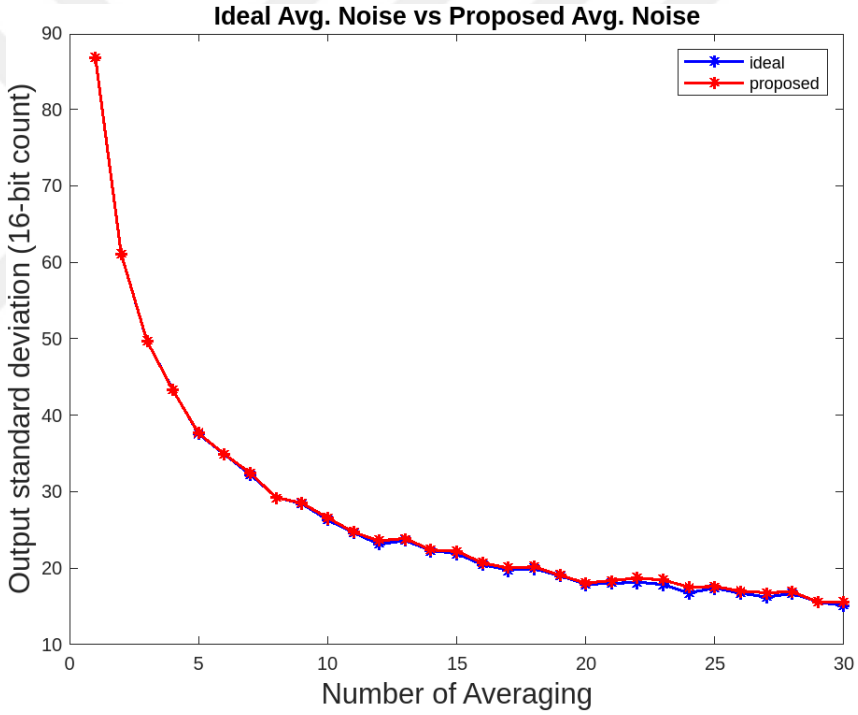


Figure 2.5. Simulation results of ideal averaging vs the proposed method. Blue line shows the standard deviations of the results obtained by the ideal calculation, red line shows the standard deviations of the results obtained by the proposed method.

The proposed method is proved qualified enough to suppress some of the intrinsic noises found in the CMOS infrared imagers regarding the results of this preliminary work. The proposed method requires only storing the coefficient values and old averaging results in memory, and since every new averaging value for a pixel is calculated as pixels are streaming, all calculation requires less logic utilization than the ideal implementation. Considering these resource advantages, on-chip averaging can be implemented using this method.

Finally, the reduced frame rate can be considered as a drawback of averaging in the CMOS infrared imagers at the first glance. However, in some of the applications like human body temperature measurement, high frame rates are traded off with low noise-high SNR frame outputs, and on-chip averaging is well suited to such applications.

2.5 Dead Pixel Correction

Dead pixels are caused by the deformed pixel structures, and they become highly distinguishable as the total number of pixels in the FPA gets lower. Due to their striking negative visual contribution, the dead pixels should be corrected before releasing the final image.

Correction of the dead pixels in the final image is very common practice. In one study, the dead pixels are corrected with the median of the neighbor pixel values [21]. Another work utilizes the nearest neighborhood algorithm [22]. Some of the methods suggest the detection of the dead pixels first and correcting the values of the dead pixels using analog comparators within the readout stage [23].

In the proposed ASIC, the dead pixel correction method is applied on-chip. Dead pixel detection is leaved to outside of the chip due to the limited layout area and tight power constraints. Accompanying CMOS infrared detector to ASIC circuit has 16x16 pixels which is relatively small in array size. After the mems process of this detector, more than 99% pixel yield is expected. In this case, 2 dead pixels

corresponds to 0.78% of the full array size. Thus, the proposed method has correction capability up to 4 dead pixels.

The suggested method does not include the identification of the dead pixels and these dead pixels are determined outside of the chip. Row and column information of the dead pixels are stored inside the ASIC during runtime. Depending on the dead pixel location, algorithm applies one of the three correction schemes. The first correction scheme is applied if the dead pixel location is inside the FPA. The second correction scheme is applied if the dead pixels are on the edges of the FPA excluding the corners. The third correction scheme is applied if the dead pixels are the corner pixels of the FPA.

The first correction scheme simply replaces the dead pixel value by the average of the 8 neighbor pixel values. Figure 2.6 depicts the possible dead pixel locations where the first correction scheme is applied.

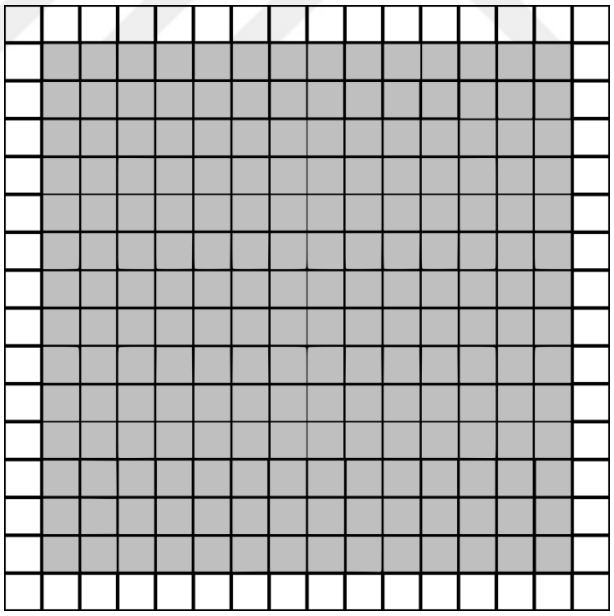


Figure 2.6. Possible dead pixel locations where the first correction scheme is applied.

Pixels among the FPA which are not located on the edges are the target of the first scheme.

Figure 2.7 shows the pixel designations used in the first scheme. Equation for the first scheme is,

$$P_c = \frac{1}{8}\sum_{n=0}^{7} P_n \tag{2.5.1}$$

P_0	P_1	P_2
P_7	P_c	P_3
P_6	P_5	P_4

Figure 2.7. Pixel designations used in the first scheme.

The second correction scheme replaces the dead pixel value by the weighted average of the 5 neighbor pixel values. In the calculation, the values of the orthogonal neighbors of the dead pixel are multiplexed by 2, and the values of remaining two pixels are leaved as they are. Then these values are summed up and divided by 8. Figure 2.8 depicts the possible dead pixel locations where the second correction scheme is applied.

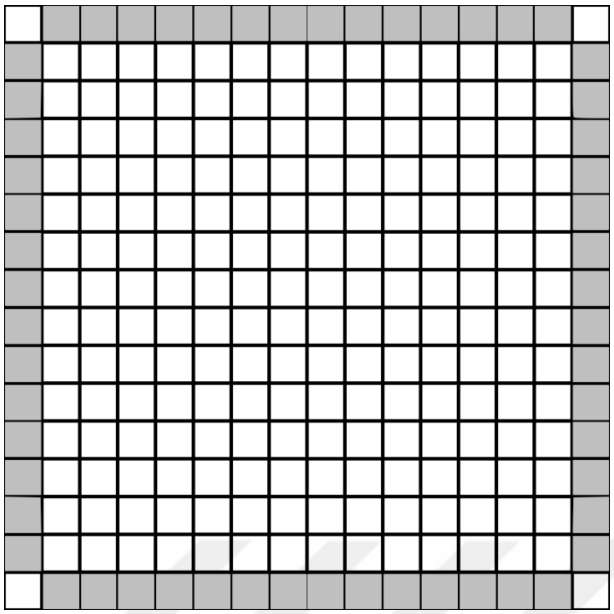


Figure 2.8. Possible dead pixel locations where the second correction scheme is applied.

There are 4 possible location groups for the second scheme, top-side, right-side, bottom-side and left-side. The algorithm is branched out to 4 for each location group. The first location group for the second scheme is the top-side group. Figure 2.9 shows the pixel designations for the top-side group. Equation for the top side is,

$$P_c = \frac{(2\,.P_3 + P_4 + 2\,.P_5 + P_6 + 2\,.P_7)}{8} \qquad (2.5.2)$$

P_0	P_1	P_2
P_7	P_c	P_3
P_6	P_5	P_4

Figure 2.9. Pixel designations for the top-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array.

The second location group for the second scheme is the right-side group. Figure 2.10 shows the pixel designations for the right-side group. Equation for the right side is,

$$P_c = \frac{(2\,.P_5 + P_6 + 2\,.P_7 + P_0 + 2\,.P_1)}{8} \qquad (2.5.3)$$

P_0	P_1	P_2
P_7	P_c	P_3
P_6	P_5	P_4

Figure 2.10. Pixel designations for the right side group of the second scheme. Pixels with darker grey color are not part of the focal plane array.

The third location group for the second scheme is the bottom-side group. Figure 2.11 shows the pixel designations for the bottom-side. Equation for the bottom side is,

$$P_c = \frac{(2\,.P_7 + P_0 + 2\,.P_1 + P_2 + 2\,.P_3)}{8} \tag{2.5.4}$$

P_0	P_1	P_2
P_7	P_c	P_3
P_6	P_5	P_4

Figure 2.11. Pixel designations for the bottom-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array.

The fourth location group for the second scheme is the left-side group. Figure 2.12 shows the pixel designations for the left-side group. Equation for the left side is,

$$P_c = \frac{(2\,.P_1 + P_2 + 2\,.P_3 + P_4 + 2\,.P_5)}{8} \tag{2.5.5}$$

P_0	P_1	P_2
P_7	P_C	P_3
P_6	P_5	P_4

Figure 2.12. Pixel designations for the left-side group of the second scheme. Pixels with darker grey color are not part of the focal plane array.

The third correction scheme is used in cases where there are three neighbor pixel values in the vicinity of the dead pixel. Figure 2.13 depicts the possible dead pixel locations where the third correction scheme is applied. In this case, the dead pixel value is replaced by the weighted average of these three neighbor pixel values. This time the values of the orthogonal neighbors to the dead pixel are multiplexed by 3, and the value of the remaining pixel is multiplexed by 2. Then, these values are summed up and divided by 8.

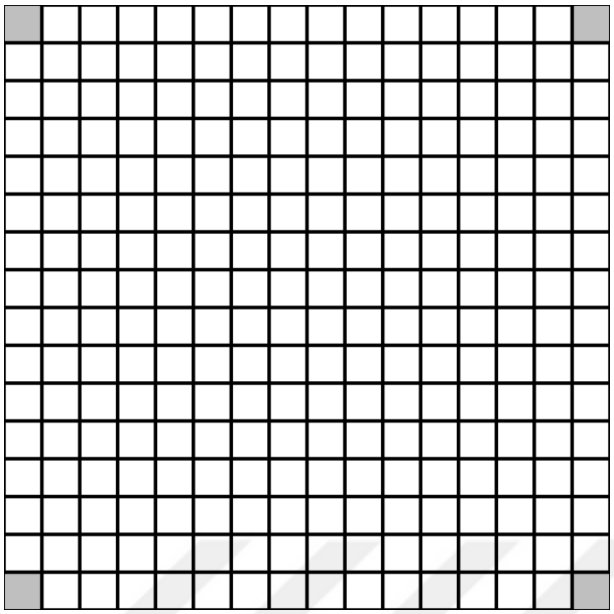


Figure 2.13. Possible dead pixel locations where the third correction scheme is applied.

As in the second scheme, there are 4 possible dead pixel locations for the third scheme, top-left corner, top-right corner, bottom-right corner and bottom-left corner. The algorithm is branched out to 4 again for each corner. The first corner pixel for the third scheme is the top-left pixel. Figure 2.14 shows the pixel designations for the top left-corner. Equation for the top left corner is,

$$P_c = \frac{(3 \,.\, P_3 + 2 \,.\, P_4 + 3 \,.\, P_5)}{8} \tag{2.5.6}$$

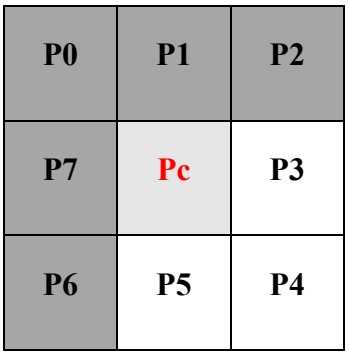


Figure 2.14. Pixel designations for the top-left corner of the third scheme. Pixels with darker grey color are not part of the focal plane array.

The second corner pixel for the third scheme is the top-right pixel.

Figure 2.15 shows the pixel designations for the top-right corner. Equation for the top right corner is,

$$P_c = \frac{(3\,.P_5 + 2\,.P_6 + 3\,.P_7)}{8} \tag{2.5.7}$$

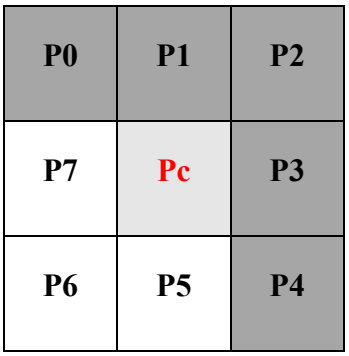


Figure 2.15. Pixel designations for the top-right corner of the third scheme. Pixels with darker grey color are not part of the focal plane array.

The third corner pixel for the third scheme is the bottom-right pixel. Figure 2.16 shows the pixel designations for the bottom-right corner. Equation for the bottom-right corner is,

$$P_c = \frac{(3 \,.\, P_7 + 2 \,.\, P_0 + 3 \,.\, P_1)}{8} \tag{2.5.8}$$

P0	**P1**	**P2**
P7	**Pc**	**P3**
P6	**P5**	**P4**

Figure 2.16. Pixel designations for the bottom-right corner of the third scheme. Pixels with darker grey color are not part of the focal plane array.

The last corner pixel for the third scheme is the bottom-left pixel. Figure 2.17 shows the pixel designations for the bottom-left corner. Equation for the bottom left corner is

$$P_c = \frac{(3 \,.\, P_1 + 2 \,.\, P_2 + 3 \,.\, P_3)}{8} \tag{2.5.9}$$

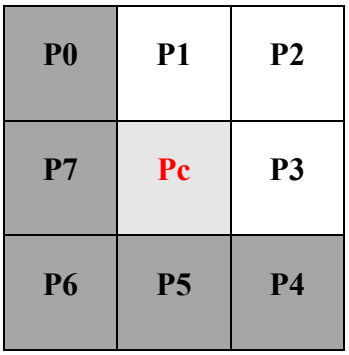


Figure 2.17. Pixel designations for the bottom-left corner of the third scheme. Pixels with darker grey color are not part of the focal plane array.

CHAPTER 3

CIRCUIT DESIGN AND IMPLEMENTATION

3.1 Introduction

This chapter presents circuit implementation of the image correction approaches on the real chip. Section 3.2 explains top level architecture of the system designed. Section 3.3 gives the details of serial interface module. Section 3.4 describes the controller unit and its sub-functions. Section 3.5 introduces the three different memory blocks to configure the chip, to store an image frame and to hold the calibration values.

3.2 Top Level Architecture

This section explains the building blocks of the designed ASIC. The main tasks of the designed ASIC are to drive the other blocks like focal plane array (FPA), pixel front end (PFE) and analog to digital converter (ADC) in order to obtain the image, and to apply on-chip image correction algorithms on the image data. The designed ASIC consists of a serial interface module, a controller unit, a configuration memory, an image memory, and a calibration memory. 5 MHz main clock input to the ASIC is created by an oscillator module which is designed by Mikrosens. Figure 3.1 illustrates the architecture of the presented ASIC design along with the other CMOS infrared imager blocks, and Figure 3.2 shows the top-level layout view. Table 3.1 summarizes the final ASIC design specifications. Table 3.2 summarizes the eminent features of each sub-block, where each sub-block of the ASIC is explained in detail in the further sections.

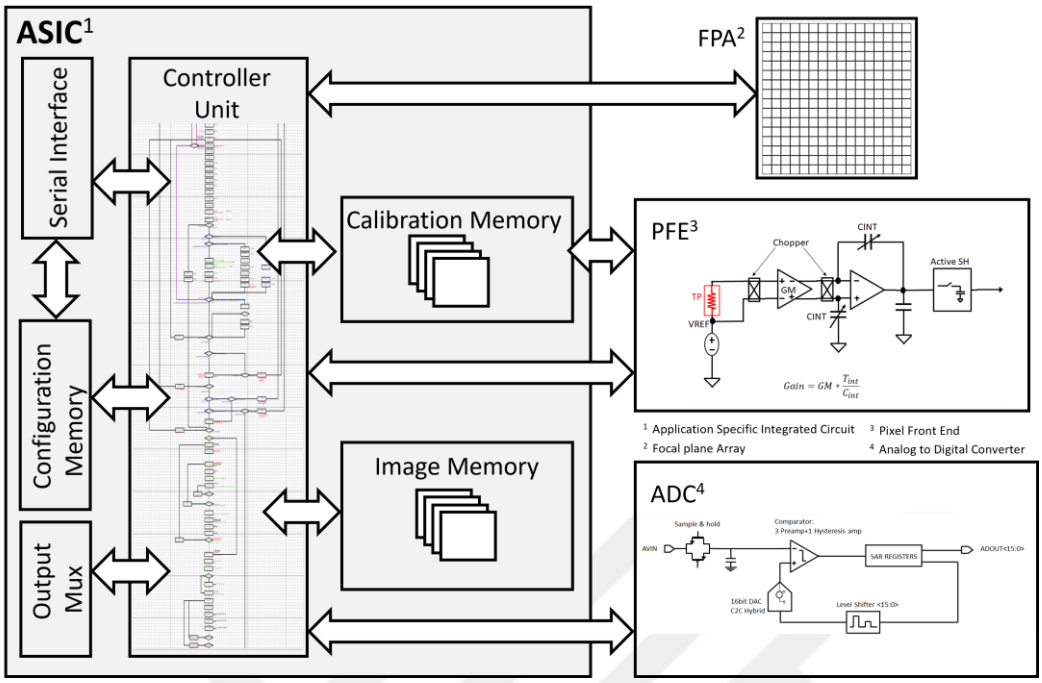


Figure 3.1. Top level architecture of the ASIC along with the other major blocks.

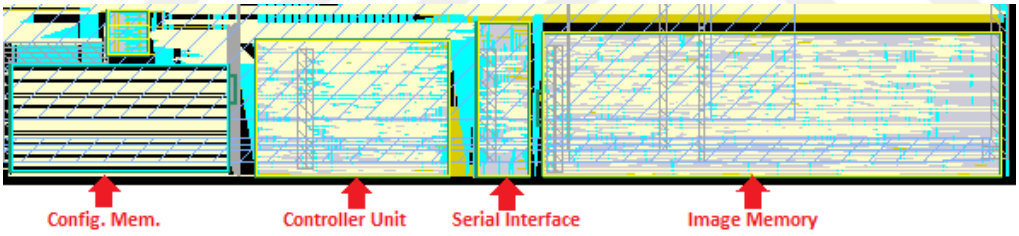


Figure 3.2. Top level layout of the ASIC. The total layout area of the circuit is 1233043 μm^2 in 0.18 µm CMOS technology.

Table 3.1. ASIC design specifications.

ASIC Specification	Value
Technology	0.18 µm CMOS
Operating Voltage	1.8 V
Max Clock Frequency	5 MHz
Total Power Consumption	2.302 mW
Total Layout Area	1233043 μm^2
Total Logic Cells	40933

Table 3.2. Eminent features of the sub-blocks of the ASIC.

ASIC Sub-block	Feature
Serial Interface	• 1 MHz SPI and I2C peripherals • Serial video output • I2C Fast Mode Plus (FM+) compatible • Soft End of Conversion (EOC) capability
Controller Unit	• 5 MHz operation • Image generation and control • Automatic readout gain calibration • Digital correlated double sampling • On-chip averaging • Dead pixel correction
Configuration Memory	• 63x16-bits, random access memory
Image Memory	• 17x16x16-bits, random access memory
Calibration Memory	• 16x20x7-bits, random access memory

3.3 Serial Interface

The serial interface module conducts the communication with outside world of the ASIC. The Serial interface module sends and receives the serial data from outside of the ASIC and handles some of the internal data transfer from the configuration memory, image memory, and calibration memory. It also steers the state machine flow of the controller unit by interpreting some of the serial data as commands. Serial interface module is composed of a core module which implements serial peripheral interface (SPI) and inter integrated circuit (I2C) peripheral hardware, a memory selection decoder, a 16-bit command memory which is reserved for instructions, and another 16-bit memory for storing soft end of conversion result. Serial interface has both SPI and I2C peripheral hardware implemented on-chip. One of these two interfaces can be selected with a selection pin. The serial interface has three modes of operation, independently of SPI or I2C selection.

The First operation mode is single word write operation. In this mode, the serial interface accepts 32-bit serial data, whose first 16-bits are address, and the remaining 16-bits are data in this mode. After receiving one serial package, serial interface module deserializes the address and data and writes this parallel data to appropriate memory. First two MSBs of the address are memory selection bits. Configuration of these two bits selects one of the 4 internal memories command memory, configuration memory, image memory or calibration memory as illustrated in Table 3.3. The command memory placed within the serial interface module whereas the other 3 memories are designed as independent modules. Content of the command memory is interpreted as commands and these commands directs the state machine flow of the controller unit like *START_IMAGING*, *OPERATION_MODE* etc.

Table 3.3. Memory address decoder bits and corresponding memories.

Address[15:14]	Selected Memory
00	Command Memory
01	Configuration Memory
10	Calibration Memory
11	Image Memory

The second mode is single word read operation. In this mode, the serial interface fetches the content of the memory from decoded address. Then, this data is serialized and outputted through one of SPI or I2C peripherals. This mode allows to read the contents of the memories to check whether the correct word is written in critical registers.

The third mode of operation is video output mode. In this mode, the content of the image memory is outputted over SPI or I2C peripherals using a relatively long, one image frame per package scheme. Sending the image data through the common protocols like SPI or I2C eases the integration of ASIC within a container system and reduces the system cost since it removes the need for relatively expensive parts like frame grabber etc. This also lowers the pin count thus more compact system designs can be achieved.

One of the implemented serial communication protocols in the serial interface module is the SPI. Implemented hardware is an SPI slave and designed to operate at 1 MHz clock speed. Four signal pins are dedicated for SPI. These pins go directly from the serial interface core to pad-frame of the chip. Master out slave in (MOSI) pin carries the data from master to slave. Master in slave out (MISO) pin carries the data from slave to master. Serial clock (SCK) pin carries the clock generated by master. Chip select (CS) pin functions as an active low device selection. Functions of these pins are summarized in Table 3.4. The SPI peripheral circuit is designed to comply with SPI standard configurations CPOL:0 and CPHA:0 (SPI Mode:0) which

means that the circuit latches the serial data at negative edges of the SCK, and it changes the content of the output shift register at positive edges of SCK when streaming the data output. In addition, it accepts a low to high transition as first SCK event.

Table 3.4. Pin functions of the SPI peripheral circuit.

Pin Name	Function
MISO	Master in slave out
MOSI	Master out slave in
SCK	Serial clock
CS	Chip select

In the single word write mode, 32-bits are transferred in one package. Simulation results of an SPI data package for single word write mode is shown in Figure 3.3. In this mode, SPI operation is started by pulling the CS line from high to low by master. SPI circuit catches the CS transition and clears the reset of internal SPI receive shift register. During the transmission, MOSI line is sampled by receive shift register at every negative edge of SCK for 32 clocks. SPI operation is finished by the master by pulling the CS line high.

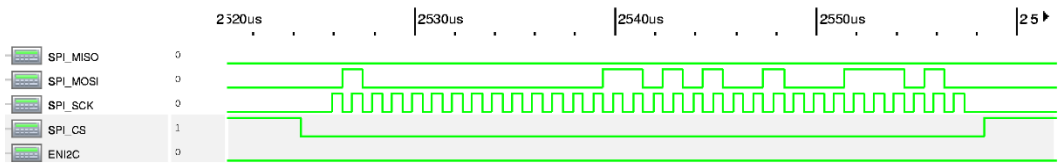


Figure 3.3. Simulation results of an SPI data package for single word write mode.

In the single word read mode, the serial interface module should be configured to SPI read mode. In order to do that, first *SPI_READ_MODE* command has to be written to the command memory. After that, two consecutive SPI frames are transmitted. In the first frame, address of the memory whose content is queried is decoded. In the second frame, the serial interface loads the content of this memory to transmit buffer and outputs it serially through MISO line. The simulation results of the SPI single word read operation is illustrated in Figure 3.4.

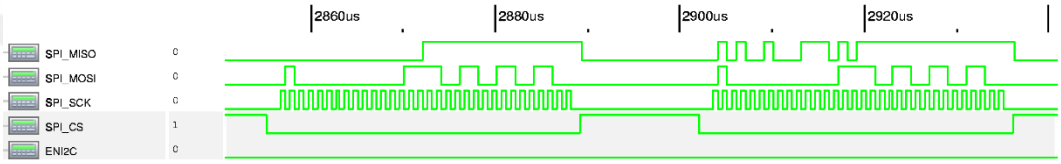


Figure 3.4. Simulation results of SPI data packages for single word read mode.

In the video output mode, 17x16x16 bits (one frame of image plus auxiliary data) is transferred from the serial interface as a one big package as shown in Figure 3.5. CS line is pulled low by the master as it is done in the other modes, and video package transmission is started. During the transmission, the serial interface requests a new pixel data every time after transmitting 16 bits (one pixel), and controller unit responds this request by fetching a new pixel data from image memory and loading it into video buffer of the serial interface. A custom handshake mechanism is established in order to avoid any synchronization issue between the serial interface module and controller unit since these modules work at different clock domains.

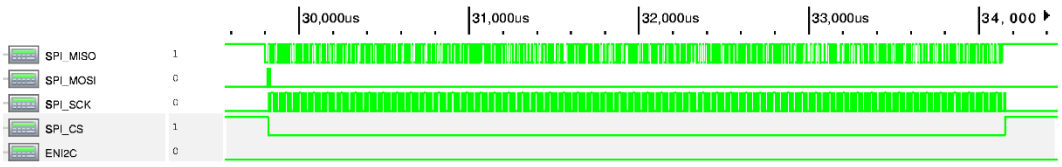


Figure 3.5. Simulation results of video output mode of SPI.

The custom handshake mechanism is demonstrated in Figure 3.6. In the figure, the serial interface requests new pixel data by asserting *REQ_RAM_DATA* signal. The controller unit fetches the next pixel data from the image memory and loads it to the serial interface's video transmit buffer with *LOAD_RAM_DATA* signal. Once the *LOAD_RAM_DATA* signal is detected, the serial interface de-assets *REQ_RAM_DATA* signal and asserts *RAM_DATA_LATCHED* signal. This time, the controller unit detects the *RAM_DATA_LATCHED* signal and de-asserts the *LOAD_RAM_DATA* signal. At the falling edge of the *LOAD_RAM_DATA* signal, the serial interface de-asserts the *RAM_DATA_LATCHED* signal and initiates the serial transmission of the new pixel data after the previous one.

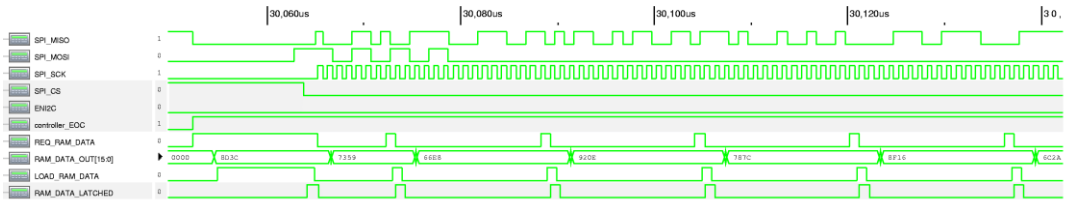


Figure 3.6. Simulation results of the custom handshake mechanism.

The other common serial communication protocol, I2C, is also implemented in the serial interface module in order to increase compatibility with diverse systems. Implemented hardware is an I2C slave and it is designed to operate at 1 MHz clock speed which compatible with I2C fast mode plus (FM+) [24]. I2C device address is

designed to be 7-bit and 5 most significant bits are fixed within the hardware as “10110”, whereas the remaining least significant two bits are selectable by the external firmware. 4 pins are reserved for I2C, serial data (SDA), serial clock (SCL), address pin 1 (A1), and address pin 0 (A0). As in SPI, these pins go directly from the serial interface core to pad-frame of the chip. SDA is the bi-directional data pin which carries the data both from the master to ASIC and from ASIC to the master. SCL is the clock pin which synchronizes the data transfer, and it is generated by the master. A1 and A0 are the two least significant bits of the 7-bit I2C device address where A0 is the least significant bit. Functions of these pins are summarized in Table 3.5. Unlike the SPI pads, I2C pads are open drain type which allows SDA and SLC lines to be pulled up by external resistors so that these lines remain high when there is no active data transmission.

Table 3.5. Pin functions of the I2C peripheral circuit.

Pin Name	Function
SDA	Serial data
SCL	Serial clock
A1	6th bit of 7-bit I2C slave address
A0	LSB of 7-bit I2C slave address

Every I2C package consists of 8-bit and a followed by an acknowledgement (ACK) bit. ACK bit is generated by the receiver of a package which can be either master or ASIC depending on the direction of the transmission. Thus, control of the SDA line switches between the master and slave. The three operation modes of the serial interface explained before also apply to I2C. Figure 3.7 explains the single word write operation using I2C. In single word write mode, the transmission is started by the master by pulling SDA line high to low while SCL line is high. This is known as the I2C start condition. During the transmission, new data is asserted when SCL is low. The transmission is finished by the master by pulling SDA line low to high while SCL line is high. This is known as the I2C stop condition.

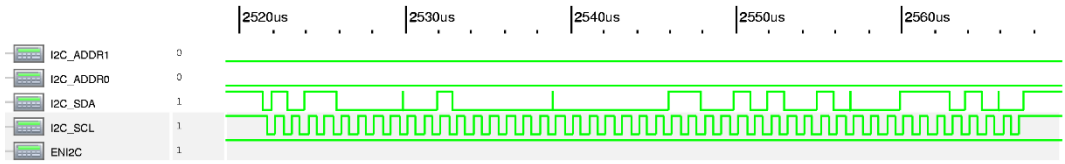


Figure 3.7. Simulation results of an I2C data package for single word write mode.

A write sequence followed by a read sequence is required in the single word read operation. Figure 3.8 shows the simulation results for single word read operation in I2C in detail. When I2C is selected, there is no need to put the serial interface to a special read mode as in SPI. Master starts the transmission by applying the start

condition and sends the device address with a following write bit. Then master sends address of the memory which is queried. After that, master asserts the start condition again without asserting a stop condition and this is known as I2C repeated start condition. Master sends device address again but this time with a following read bit. After the read bit, control of the SDA line is transferred to the ASIC. The serial interface outputs the content of the memory whose address is decoded in previous package. Finally, the master applies a stop condition and hence ends the transmission.

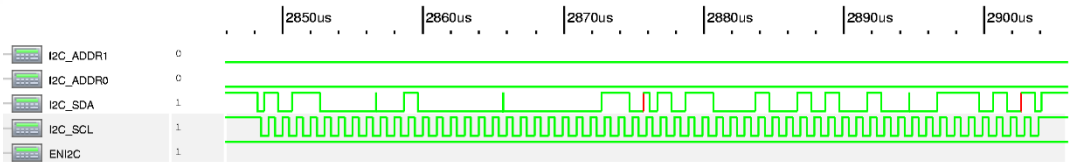


Figure 3.8. Simulation results of an I2C data package for single word read mode.

The video output mode is similar to single word read mode. In the video output mode, ASIC sends the whole content of the image memory instead of the content of a single register. This can be seen in the simulation results for video output mode in I2C as illustrated in Figure 3.9. Since every pixel is represented by 16-bit data, outputting a pixel takes two consecutive 8-bit transmissions. Video output transmission is initiated by applying a start condition by the master, last for 272 pixels and ends with a stop condition by the master. Similar handshake mechanism as in the SPI is used in this transmission.

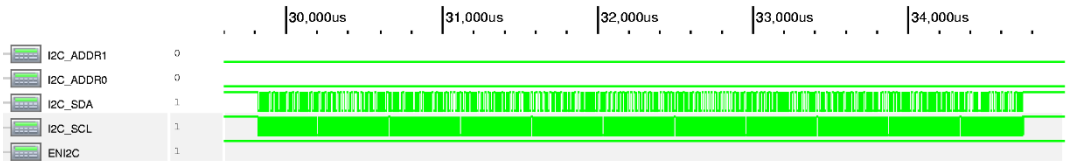


Figure 3.9. Simulation results of video output mode of I2C.

The serial interface module is designed, and the layout is implemented by cutting edge EDA tools. Schematics for the serial interface module is designed by schematic capture method, and layout is generated by automatized layout generation tools. Figure 3.10 shows the final layout view of the design. Proper constraints are determined and given as inputs to the layout generation tool. Blank spaces are filled with filler cells and decap cells in order to satisfy the metal and doping uniformity which is essential for CMOS manufacturing. Total logic utilization, power consumption estimation and layout area of the final design of the serial interface module are given in Table 3.6.

Table 3.6. Final design specifications of the serial interface module.

Specification	**Value**
Pin Count	173
Layout area	404.3 µm x 134.4 µm
Total Power Consumption	0.042 mW
Clock frequency	1 MHz
Number of logic cells	2253

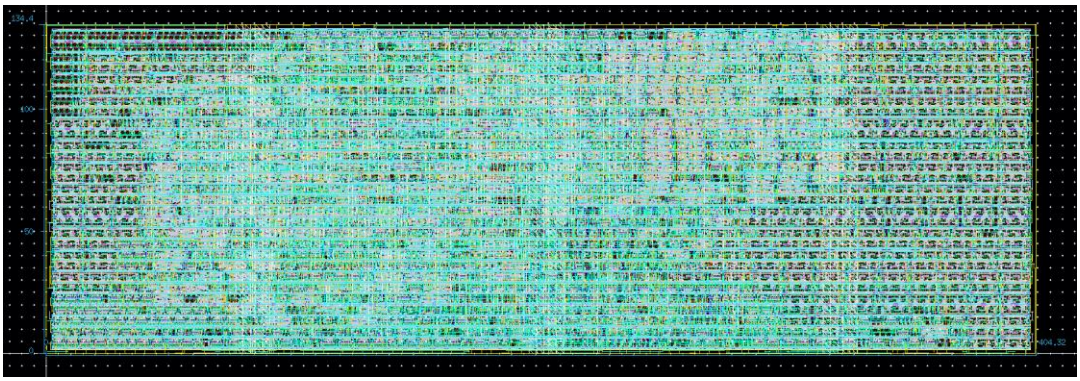


Figure 3.10. Final layout view of the serial interface module. The total layout area of the circuit is 404.3 µm x 134.4 µm in 0.18 µm CMOS technology.

3.4 Controller Unit

3.4.1 Introduction

The controller unit is the main component of the ASIC which orchestrates the whole data generation and flow within the chip. It generates necessary timing signals to other blocks such as PFE and ADC. The controller unit also manages the storing of the pixel values in image memory and synchronizes the video output. In addition, the controller unit applies on-chip image correction algorithms to the image. The controller unit comprises of a top state machine and a core state machine. The top state machine controls the flow of the main modes of the ASIC such as imaging mode, calibration mode, test mode etc. The core state machine, on the other hand, generates timing signals required to stream the image properly, drives other blocks, handles memory interactions, generates test signals, and most importantly apply the image correction algorithms on streaming image. Figure 3.11 describes the generalized flow diagram of the top state machine. Main functions of the implemented controller unit circuit can be grouped into five subtitles: image generation, automatic readout gain calibration, correlated double sampling, on-chip

averaging, and dead pixel correction. Further sections will give detailed explanation on these functions as well as circuit implementation details.

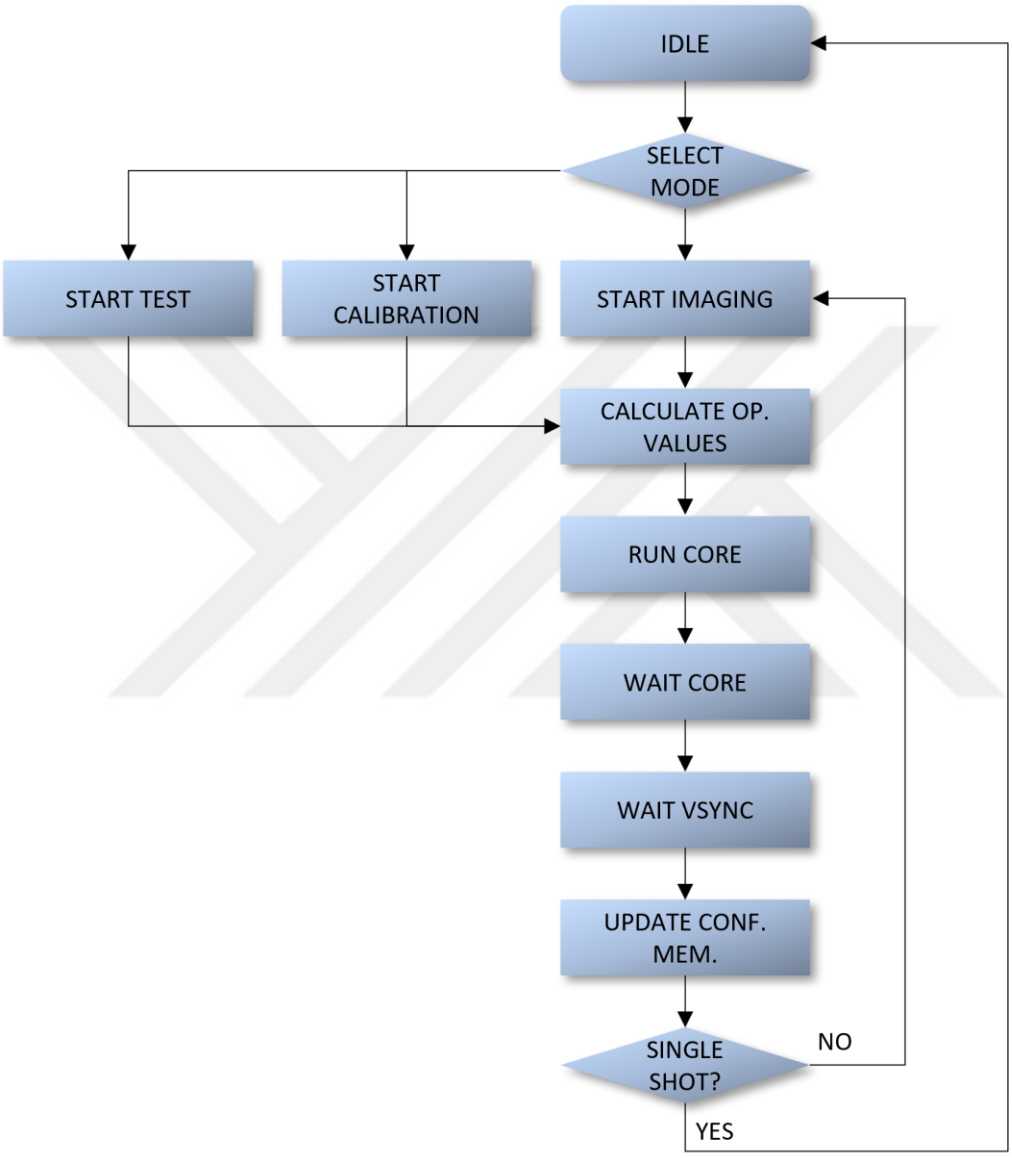


Figure 3.11. Flow diagram of the top state machine of the controller unit.

3.4.2 Image Generation

The infrared image frame is basically a pile of pixel values in the focal plane array that lined up in the form of rows and columns. In order to establish a meaningful image stream, pixel values should be collected, processed and sorted in a harmony. Driving the PFE and ADC and sorting the infrared image data constitutes the basis of the image generation function of the controller unit.

The controller unit utilizes 2x5-bit *row_counter* and *col_counter* counters to conduct the image generation operation. In the configuration memory 4x5-bit registers, the *RG_ROW_BEGIN* and *RG_COL_BEGIN* registers hold the beginning row and column values whereas *RG_ROW_END* and *RG_COL_END* registers hold the row and column reading limits. This feature of the ASIC gives user flexibility to read selected area within the FPA. The content of these registers, and hence the reading area among the FPA, can be changed later by changing the content of the certain addresses of the configuration memory. Outputs of the image generation function include 2x16-bit *row_select* and *col_select* buses which are routed to the FPA, and various signals which are routed to the PFE and ADC circuits. In addition, an address and data path lays between the controller unit and the memory blocks.

Algorithm 1 shows the general flow of the image generation function. After power up, the controller unit initializes *row_counter* and *col_counter* counters from the configuration memory. The controller unit adds 2 to begin value of the *col_counter* to avoid reading of the first two dummy columns. Image generation is started with *START_IMAGING* command from the serial interface. The controller unit asserts the *row_select* signal for the beginning row, hence pixels in the beginning row are connected to the PFE. Then the controller unit generates the required PFE waveforms. These waveforms run the PFE circuit, and in consequence small infrared signals from the detector pixels are amplified. After amplification, the controller unit asserts *col_select* signal for the beginning column. This connects the output of the beginning column PFE circuit to the ADC. After selecting the column, ADC waveforms are generated, and hence infrared signal for one pixel is converted to

digital domain. Following the conversion, digital pixel value is accepted by the controller unit for further processing. Various correction methods are applied to the pixel value. Final pixel value then stored in the image memory. At this point, the controller unit increases the value of the *col_counter* by 1 and repeats the operation until *RG_COL_END* limit is reached but without generating waveforms for PFE and ADC. After the column limit is reached, the controller unit increases *row_counter* by one and repeats the whole operation until *RG_ROW_END* limit is reached.

The image generation is the backbone function of the controller unit, and it runs whenever imaging is in question. Image correction algorithms run inside the image generation flow. After filling the image memory for one frame, controller unit asserts end-of-conversion (*EOC*) signal to outside the chip. This EOC signal tells the external firmware image is ready to be streamed out through the serial interface.

Algorithm 1 Image Generation

Initialize

$row_counter \leftarrow RG_ROW_BEGIN$
$col_counter \leftarrow RG_COL_BEGIN + 2$
$row_select \leftarrow 0x0001$
$col_select \leftarrow 0x0001$

while $(row_counter < (RG_ROW_END + 1))$ **do**
    $row_select \leftarrow (row_select << (row_counter - 1))$
    $< pfe_operations... >$
    while $(col_counter < (RG_COL_END + 3))$ **do**
        $col_select \leftarrow (col_select << (col_counter - 3))$
        $< adc_operations... >$
        $< im_corr_operations... >$
        $col_counter++$
    $row_counter++$
$EOC \leftarrow 1$

Algorithm 1. Image generation.

Simulation results of the image generation function is shown in Figure 3.12 and Figure 3.13 below. As shown in the first figure, as *row_counter* increases corresponding *row_select* signal is asserted. The second figure shows a row reading time in detail. In each row reading time, *col_counter* is increased and corresponding *col_select* signal is asserted. In the simulation, begin and end limits are configured to minimum and maximum limits to show whole frame scanning.

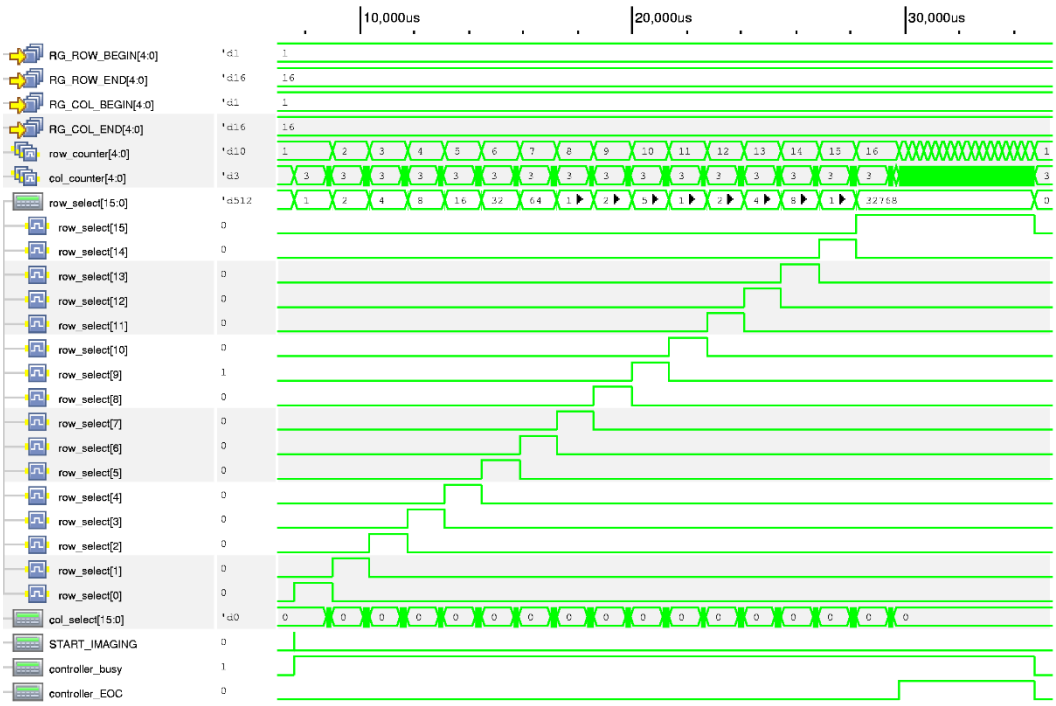


Figure 3.12. Simulation results of the image generation. Rows are selected one by one. After all row reading is finished, infrared image for one frame is stored in the image memory and this is flagged to the external firmware by EOC bit.

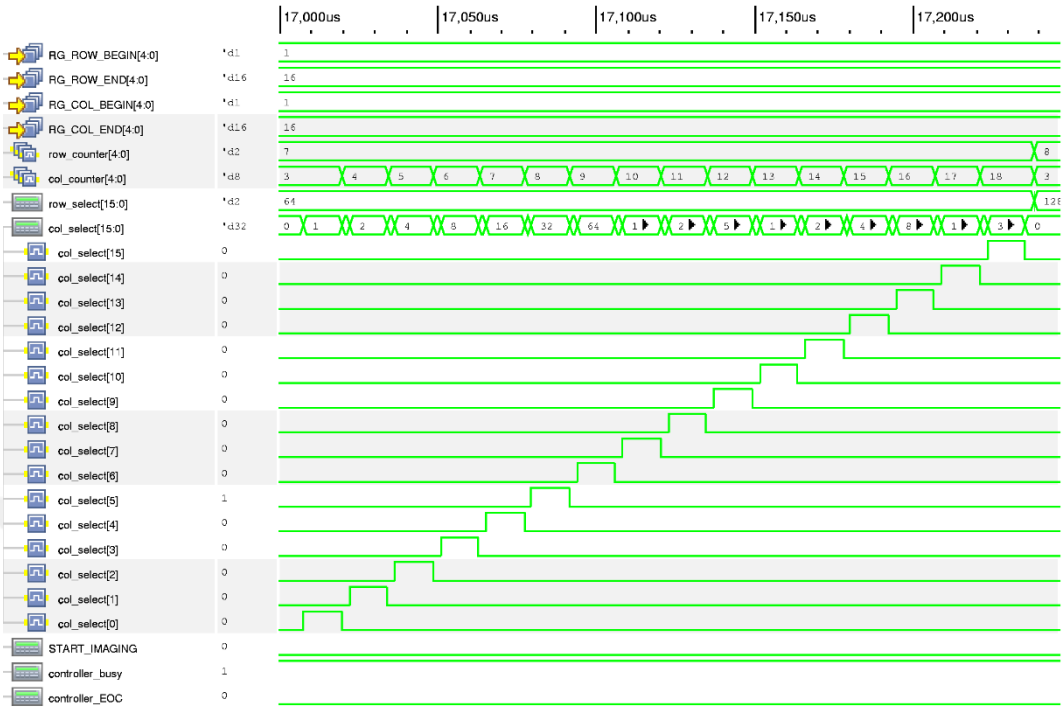


Figure 3.13. Simulation results of the image generation. Columns are selected one by one inside a row reading time.

3.4.3 Automatic Readout Gain Calibration Implementation

The automatic redout gain calibration is the first image correction method implemented in the ASIC, among the four methods. The algorithm explained in the section 2.2.2 is implemented, and the hardware for the algorithm is synthesized within the controller unit module of the ASIC.

The controller unit utilizes an auxiliary 16x20x7-bits calibration memory which holds the values for the Cint array of the PFE. A 16-bit *cal_compare* register buffers converge value stored in the configuration memory. A 3-bit *cal_counter* flags the ongoing calibration stage. In addition, 2x16-bit *cal_row_select* and *cal_col_select*

registers, a 7-bit *cal_current_bit* register, and 4x1-bit *cal_toggle*, *cal_wr*, and *cal_en* registers are utilized to conduct the calibration memory operations.

Algorithm 2 shows the automatic readout gain calibration flow. Following the power up, the calibration memory is filled with all 1000000, which is the intermediate value for the capacitor array. Firstly, the controller unit should be put into calibration mode through changing the *CONTROLLER_MODE* bits to 10. This mode changes the image generation flow according to calibration. Automatic calibration operation starts with the *START_IMAGING* command. After the ADC conversion, digital pixel value from the ADC is accepted by the controller unit and stored in the *adc_data_reg* register, and 16-bit converge value stored in the configuration memory is buffered into the *cal_compare* register in the controller unit. After buffering, the adc value is compared with the *cal_compare* value. If the adc value is less than the compare value, most significant bit of the calibration memory which is set to 1 after power up is cleared. If the adc value is greater than the compare value, content of the calibration memory is leaved as it is. This operation is repeated until all the pixels in a frame is finished. When frame is finished, the controller unit increases the value of the *cal_counter* by 1 and sets all the second MSBs of the calibration memory. Comparison and memory operations are repeated for the second MSBs of the calibration memory. Automatic calibration operation is repeated until *cal_counter* value is reached to 7. Therefore, all automatic gain calibration operation is done after 7 frames.

Algorithm 2 Automatic Readout Gain Calibration

Initialize

$cal_counter \leftarrow 0$
$cal_current_bit \leftarrow 0b1000000$
$cal_compare \leftarrow RG_CAL_CONVERGE_VAL$
$cal_active \leftarrow 1$
while $(cal_counter < 7)$ **do**
    $cal_current_bit \leftarrow (cal_current_bit >> cal_counter)$
    $CAL_MEM[cal_current_bit] \leftarrow 1$
    if $(adc_data_reg < cal_compare)$ **then**
        $CAL_MEM[cal_current_bit] \leftarrow 0$
    else
        $CAL_MEM[cal_current_bit] \leftarrow CAL_MEM[cal_current_bit]$
    $cal_counter++$
$cal_active \leftarrow 0$

Algorithm 2. Automatic readout gain calibration.

The simulation results for the automatic readout gain calibration function is presented in Figure 3.14 and Figure 3.15. The first figure shows the general calibration flow as frames stream. In the figure, *cal_counter* increases as new frames arrive and *cal_current_bit* changes accordingly. *cal_toggle* bit sets the current bits of the calibration memory at the beginning of each new stage. In the second figure, details of a calibration operation within a calibration stage are shown. First incoming adc data is 0x7240 which is smaller than the compare value of 0x8000, so previously set current bit of the calibration memory for this pixel is cleared and hence the capacitance value of the readout circuit during the reading time of that pixel is shrinked from 0x48 to 0x40 as seen on the signal *CAL_Q3*. Eventually the readout gain for that pixel is increased and adc output approaches to the converge value. In the figure, the third incoming adc value is 0x8190 which is greater than the compare

value 0x8000. In this case, the controller unit does not clear the previously set current bit of the calibration memory, so no write operation happens for this pixel as seen on the signal *CAL_Q5*. Similar decisions are made for all incoming adc data and final values are stored in the 16x20x7-bits calibration memory. However, only five outputs of the calibration memory are included in the figure for the sake of clarity.

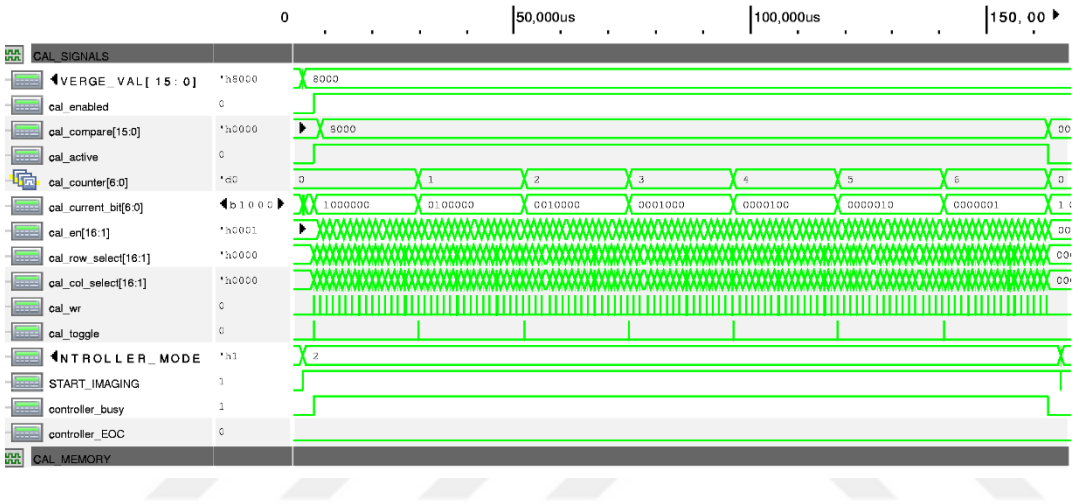


Figure 3.14. General view of the simulation results of the automatic readout gain calibration. *cal_counter* indicates the current calibration stage.

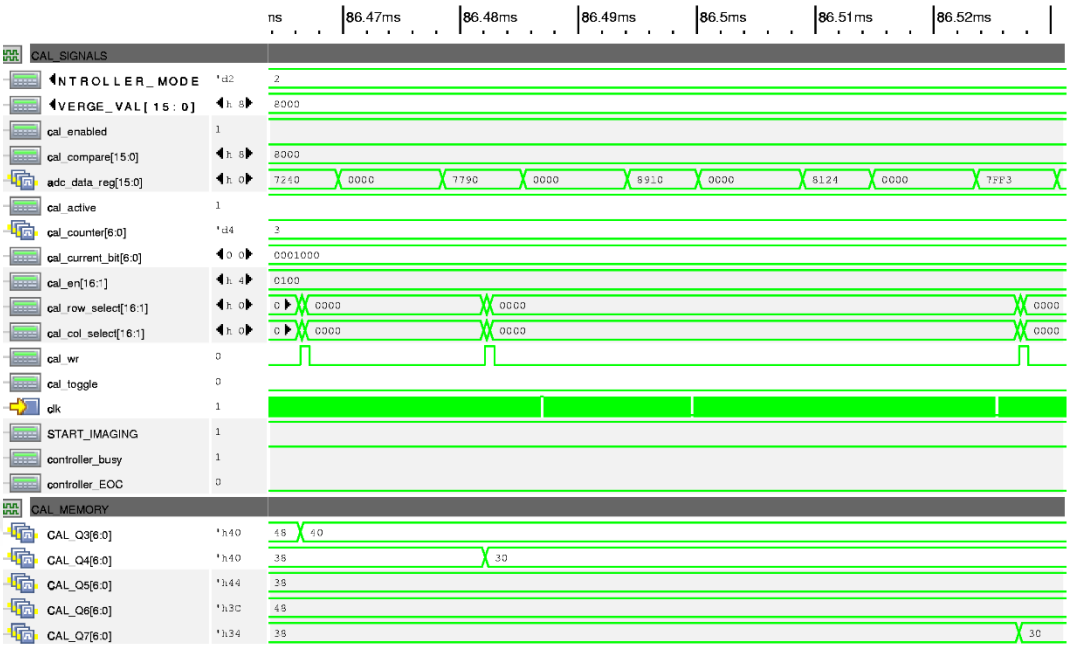


Figure 3.15. Detailed view of the simulation results of the automatic readout gain calibration.

3.4.4 Correlated Double Sampling Implementation

Another image correction method implemented in ASIC is the correlated double sampling. The method proposed in the section 2.3.3 is implemented within the controller unit. In order to increase the flexibility of the operation, an offset adder is also integrated to the CDS circuit.

The digital CDS hardware utilizes the 17x16x16-bit image memory as a temporary storage to hold integration values of the PFE. In the controller unit, a 16-bit *current_adc_data* register holds the incoming ADC data, and a 16-bit *recalled_adc_data* register holds the values of the integration values recalled from the image memory. Another 16-bit *cds_calculated* register holds the output value of the subtraction operation. A 4-bit *RG_OFFSET_VALUE* register is also included to

add artificial offset to the final pixel value. Image memory signals are used to data transmission between the image memory and the controller unit.

Algorithm 3 shows the CDS implementation steps. In order to perform the CDS, *RG_CDS_ENABLED* bit of the configuration memory is set. Firstly, PFE is put into integration state by the controller unit and *PC_READ_VINT* signal is asserted. Then integration state outputs of the PFE for one row are stored in the image memory. After that, the controller unit resets the col_counter and put PFE circuit into reset state and *PC_READ_VINT* signal is de-asserted and *PC_READ_VRST* signal is asserted. This time reset values of the PFE circuit for one row are read. When the first reset value is arrived at the controller unit, the integration state value for that pixel is recalled from the image memory and buffered in the controller unit. If offset addition function is enabled, offset value is buffered from the configuration memory and this value is added to the integration state value of the pixel. Reset value is then subtracted from the integration value and the calculated result is rewritten to the image memory. These steps are repeated until *RG_COL_END* limit is reached. After that, the controller unit increases the value of the *row_counter* by 1 and follows the same CDS steps. Operation is repeated until all rows are finished.

Algorithm 3 Correlated Double Sampling

```
1: Initialize
2: PC_READ_VINT ← 0
3: PC_READ_VRST ← 0
4: while (row_counter < row_end_limit) do
5:     while (col_counter < col_end_limit) do
6:         PC_READ_VINT ← 1
7:         PC_READ_VRST ← 0
8:         < ram_write... >
9:         col_counter + +
10:    col_counter ← 0
11:    while (col_counter < col_end_limit) do
12:        PC_READ_VINT ← 0
13:        PC_READ_VRST ← 1
14:        < ram_read... >
15:        offset_added ← recalled_adc_data + RG_OFFSET_VALUE << 12
16:        cds_calculated ← (offset_added − current_adc_data)
17:        < ram_write... >
18:        col_counter + +
19:    row_counter + +
20: EOC ← 1
```

Algorithm 3. Correlated double sampling.

The simulation outputs related to the CDS function are presented in Figure 3.16 and Figure 3.17. In the first figure, internal signals *PC_READ_VINT* and *PC_READ_VRST* indicate the integration state read time and reset state read time for a row, respectively. The offset addition and the CDS operation happen in a reset state read time. Detailed operation for one pixel is shown in the second figure. In the figure, incoming ADC data for the reset state of the readout is shown in *current_adc_data* register which is 0x451A. Integration state data for that pixel is read back from the image memory and is buffered in *recalled_adc_data* register which is 0x6C2A. Next, offset value is right shifted by 12 and added to *recalled_adc_data*. This value is stored in *offset_added* register which is 0xFC2A. Finally current_adc_data is subtracted from this value yielding *cds_calculated* data which is 0xB710. This value is written back to the image memory replacing the old integration state data for that pixel.

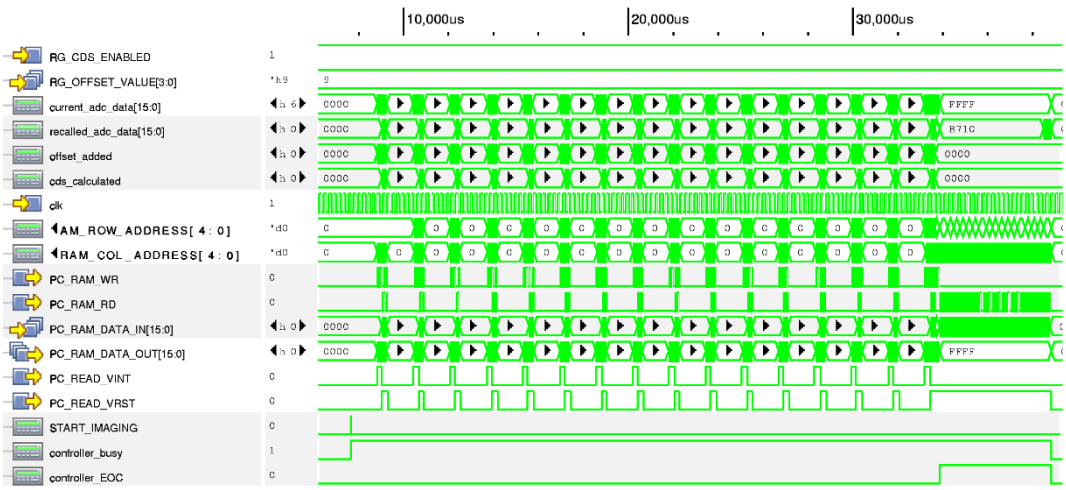


Figure 3.16. General view of the simulation results of the correlated double sampling. *PC_READ_VINT* and *PC_READ_VRST* signals indicate integration state and reset states of the readout circuit.

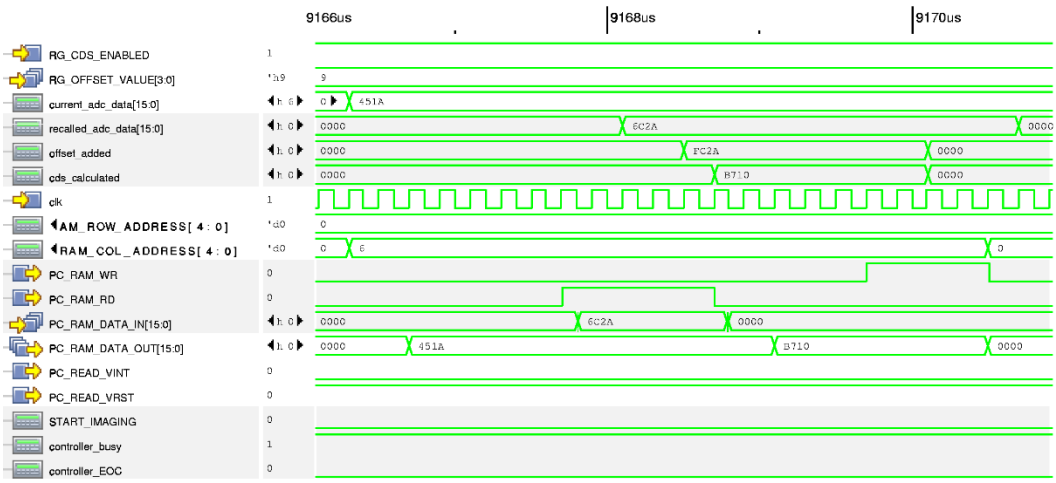


Figure 3.17. Detailed view of the simulation results of the correlated double sampling.

3.4.5 On-Chip Averaging Implementation

This section introduces the implementation of the averaging algorithm demonstrated in section 2.4. Averaging is implemented digitally on-chip within the controller unit flow.

5-bit memory area in the configuration memory, *RG_AVERAGE_CNT* decides the average limit. A 6-bit *avg_counter* indicates the current averaging stage. 2x16-bit registers *adc_data_reg* and *PC_RAM_DATA_IN_reg* holds the new data from the adc and old averaging result from the image memory. 2x20-bit registers, *avg_new_data* and *avg_old_data* hold the temporary values of the multiplication results regarding the maximum expected bits for the products. A 16-bit register *avg_calculated_data* holds the result of the sum and shift operations. Values of the coefficients *C1* and *C2* are hard coded as 5-bit values in the controller unit.

Algorithm 4 demonstrates the averaging operation. Averaging is in effect during the imaging operation when the *RG_EN_AVERAGING* bit in the configuration memory

is set and operation starts with *START_IMAGING* command. First, pixel values for the first frame are stored in image memory as normal imaging operation. When the first pixel value of the second frame arrives in the controller unit, it is multiplicated by the corresponding *C2* in the Table 2.1 which is 16. Then old averaging value is recalled from the image memory and is multiplicated by *C1* which is also 16. Then two products are summed up and right shifted by 5. Resulting value is stored in the image memory replacing the old averaging value for that pixel. Same steps are applied to every pixel in the frame. Hence this averaging stage is completed. At the end of each averaging stage, the controller unit increases the *avg_counter* by 1 and initiates the new averaging stage. All operation is repeated until *avg_counter* reaches to the average limit and as the result of the averaging operation, image memory is filled with the certain number of the averaged values of the pixels. At this point, the controller unit asserts EOC signal so that external firmware streams the averaged infrared image frame.

Algorithm 4 On-Chip Averaging

```
1: Initialize
2: while (avg_counter < RG_AVERAGE_CNT) do
3:     while (row_counter < row_end_limit) do
4:         while (col_counter < col_end_limit) do
5:             < ram_read... >
6:             avg_new_data ← adc_data_reg * C2
7:             avg_old_data ← PC_RAM_DATA_IN_reg * C1
8:             avg_calculated_data ← ((avg_new_data + avg_old_data) >> 5)
9:             < ram_write... >
10:            col_counter + +
11:        col_counter ← 0
12:        row_counter + +
13:    avg_counter + +
14: EOC ← 1
```

Algorithm 4. On-chip averaging.

The simulation results for the averaging function are illustrated in Figure 3.18. In the figure, averaging calculation operation for one pixel during the 3rd averaging stage out of a 30-stage run is shown. As shown in the figure, firstly, incoming adc data which is 0x0BB8 is buffered in *adc_data_reg* register. Then, old averaging result which is 0x02EE is fetched from the image memory and stored in *PC_RAM_DATA_IN_reg* register. These values are multiplied with corresponding *C2* and *C1* hard coded values which are 11 and 21, respectively. Multiplication results are temporarily stored in the registers *avg_new_data* and *avg_old_data* which are filled with 0x0080E8 and 0x003D86. Next, *avg_new_data* and *avg_old_data* values are summed and right shifted by 5. This result which has the value of 0x05F3 is temporarily stored in the *avg_calculated_data* register and designated as averaging result of the third averaging stage which will be used in the calculation for

the next stage. *avg_calculated_data* then, stored in the image memory replacing the result of the former averaging stage.

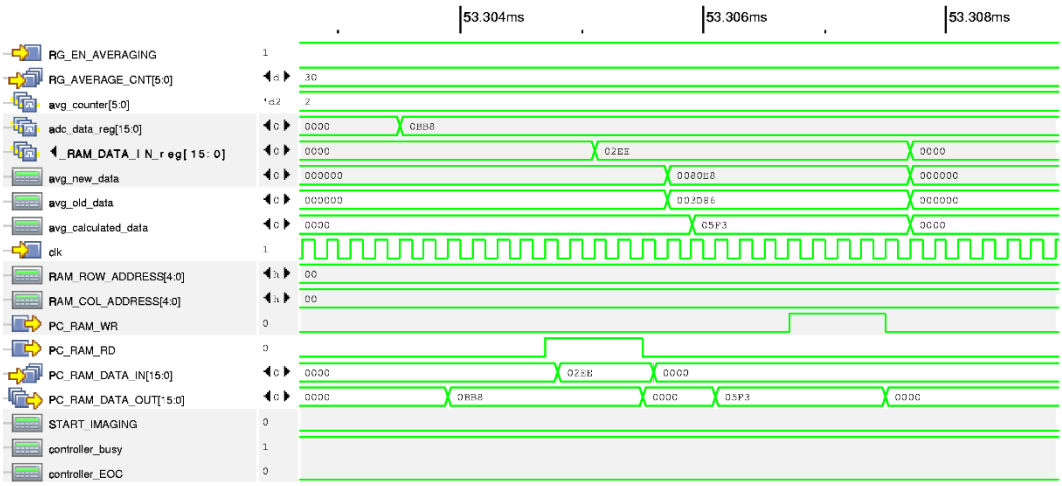


Figure 3.18. Simulation results of the on-chip averaging.

3.4.6 Dead Pixel Correction Implementation

Dead pixel correction is implemented in the controller unit flow according to the method given in section 2.5. This section gives implementation details of the dead pixel correction.

Implemented correction method assumes the dead pixel locations are identified prior to the runtime, since the detection of the dead pixels is left outside of the ASIC. Locations of the identified dead pixels are written in the configuration memory by external firmware as rows and columns. In the configuration memory, 8 x 4-bit areas are dedicated to rows and columns of the 4 dead pixels. 2-bit *d_pix_counter* indicates the current dead pixel whose correction is in progress. 2 x 4-bit registers, *current_d_pix_row* and *current_d_pix_col* hold the row and column location of the

dead pixel being corrected. A 3-bit *d_pix_neigh_counter* is utilized to identify the neighbor row and column addresses. A 3-bit *current_d_pix_neigh_end* register holds the end values of the *d_pix_neigh_counter*. A 3-bit *d_pix_neigh_repeat_counter* holds the repeat count for a neighbor pixel. A 2-bit *d_pix_neigh_repeat_value* sets the limit for the repeated sum of a neighbour value.

Algorithm 5 shows the dead pixel correction operation. The dead pixel correction operation is in action If the *RG_EN_DEAD_PIX_COR* bit is enabled in the configuration memory. First the controller unit updates the content of *current_d_pix_row* and *current_d_pix_col* registers according to the value of the *d_pix_counter*. Depending on the dead pixel location, the controller unit next decides one of the three correction schemes as indicated in section 2.5. Next, the controller unit decides one of the four dead pixel location groups if applicable. Once the correction scheme and the location group are decided, the controller unit updates the value of *current_d_pix_neigh_end* register. Next, the controller unit initializes the content of the *d_pix_neigh_counter* with the beginning neighbour number. Next, the controller unit calculates the *d_pix_neigh_repeat_value* as well as row and column address for that neighbor. After that, the controller unit fetches the content of the current neighbor from the image memory. The controller unit cumulatively sums up each new neighbor value and increases the neighbor counter by one until the *d_pix_neigh_end* limit is reached. Before increasing the neighbor counter, if repeat is required, the same neighbor value is summed up until the *d_pix_neigh_repeat_value* is reached.

Algorithm 5 Dead Pixel Correction

```
1: while (d_pix_counter < 4) do
2:     current_d_pix_row ← RG_D_PIX_ROW(d_pix_counter)
3:     current_d_pix_col ← RG_D_PIX_COL(d_pix_counter)
4:     switch (current_d_pix_row, current_d_pix_col) do
5:         case (top left corner)
6:             d_pix_neigh_counter ← 3
7:             current_d_pix_neigh_end ← 5
8:         case (top right corner)
9:             d_pix_neigh_counter ← 5
10:            current_d_pix_neigh_end ← 7
11:        case (bottom right corner)
12:            d_pix_neigh_counter ← 7
13:            current_d_pix_neigh_end ← 1
14:        case (bottom left corner)
15:            d_pix_neigh_counter ← 1
16:            current_d_pix_neigh_end ← 3
17:        case (top side)
18:            d_pix_neigh_counter ← 3
19:            current_d_pix_neigh_end ← 7
20:        case (right side)
21:            d_pix_neigh_counter ← 5
22:            current_d_pix_neigh_end ← 1
23:        case (bottom side)
24:            d_pix_neigh_counter ← 7
25:            current_d_pix_neigh_end ← 3
26:        case (left side)
27:            d_pix_neigh_counter ← 1
28:            current_d_pix_neigh_end ← 5
29:        case (all other)
30:            d_pix_neigh_counter ← 0
31:            current_d_pix_neigh_end ← 7
32:    while (d_pix_neigh_counter < current_d_pix_neigh_end + 1) do
33:        < ram_read... >
34:        switch (current_d_pix_row, current_d_pix_col) do
35:            case (corners)
36:                d_pix_neigh_rpt_value ← (d_pix_neigh_counter%2) + 1)
37:            case (sides)
38:                d_pix_neigh_rpt_value ← (d_pix_neigh_counter%2))
39:            case (all other)
40:                d_pix_neigh_rpt_value ← 0
41:        corr_pix_value ← (corr_pix_value + neigh_pix_value)
42:        while (d_pix_neigh_rpt_counter < d_pix_neigh_rpt_value + 1) do
43:            corr_pix_value ← (corr_pix_value + neigh_pix_value)
44:            d_pix_neigh_rpt_counter + +
45:        d_pix_neigh_rpt_counter ← 0
46:        d_pix_neigh_counter + +
47:    corr_pix_value ← (corr_pix_value >> 3)
48:    < ram_write... >
49:    d_pix_neigh_counter ← 0
50:    d_pix_counter + +
```

Algorithm 5. Dead pixel correction.

Figure 3.19 shows the simulation results for the dead pixel correction operation of bottom-right corner pixel which is located at 15th row and 15th column beginning with 0. In the figure, first *current_d_pix_row* and *current_d_pix_col* registers are updated according to the state of d_pix_counter. Next, initial value of *d_pix_neigh_counter* and *current_d_pix_neigh_end* values are calculated which are 7 and 1, respectively. After that, value of the first neighbor pixel is recalled from the image memory which is 0x921D. Since this neighbour pixel value is orthagonal to the dead pixel to be corrected, this value is summed up three times yielding *d_pix_corr_pix_value* 0x1B657. Then *d_pix_neigh_counter* is increased by one and the next neighbor value is fetched from the image memory which is 0x70E3. This is not an orthagonal pixel, so its value is summed up two times and added to the cumulative sum yielding *d_pix_corr_pix_value* 0x2981D. The last neighbour pixel value recalled from the image memory is 0x8E21 and it is also included three times in the sum yielding 0x44280. This final sum then right shifted by 3 yielding 0x8850. This corrected dead pixel value 0x8850 is then stored in the image memory onto the dead pixel location. Similar steps are repeated for other dead pixels.

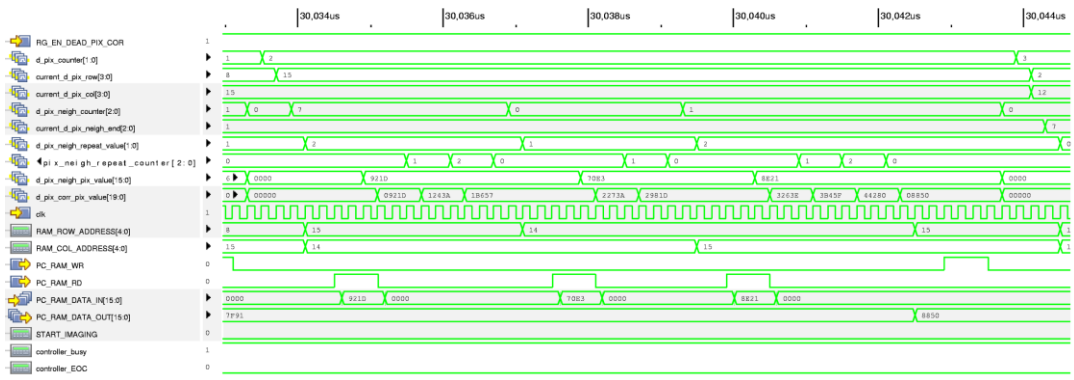


Figure 3.19. Simulation results of the dead pixel correction for the bottom-right corner pixel.

3.4.7 Controller Unit Layout

The controller unit circuit is implemented according to a full RTL to GDSII flow using the cutting-edge simulation, synthesis, and layout generation tools. RTL coding is done in Verilog HDL, and design constraints are prepared in Synopsys Design Constraints (.sdc) format. The HDL code and constraints are inputs to the synthesize tool. The generic circuit is synthesized and mapped to 0.18 µm CMOS standard cell library by the synthesize tool. Then, the synthesized code, as well as the layout design constraints are given to layout generation tool as input. Next, the synthesized circuit is placed and routed by the layout generation tool. The final layout of the controller unit is shown in Figure 3.20. Static timing analysis (STA) and estimated power analysis are done in both synthesize and layout stages. The final simulations and verifications are done including the annotated parasitic effects from the layout regarding the worst corners. The final layout area of the controller unit is 354.5 µm x 504.6 µm. Some specifications and design utilization for the controller unit is summarized in the Table 3.7 below.

Table 3.7. Final design specifications of the controller unit.

Specification	**Value**
Pin Count	468
Layout area	354.5 µm x 504.6 µm
Total Power Consumption	0.44 mW
Clock frequency	5 MHz
Number of logic cells	10268

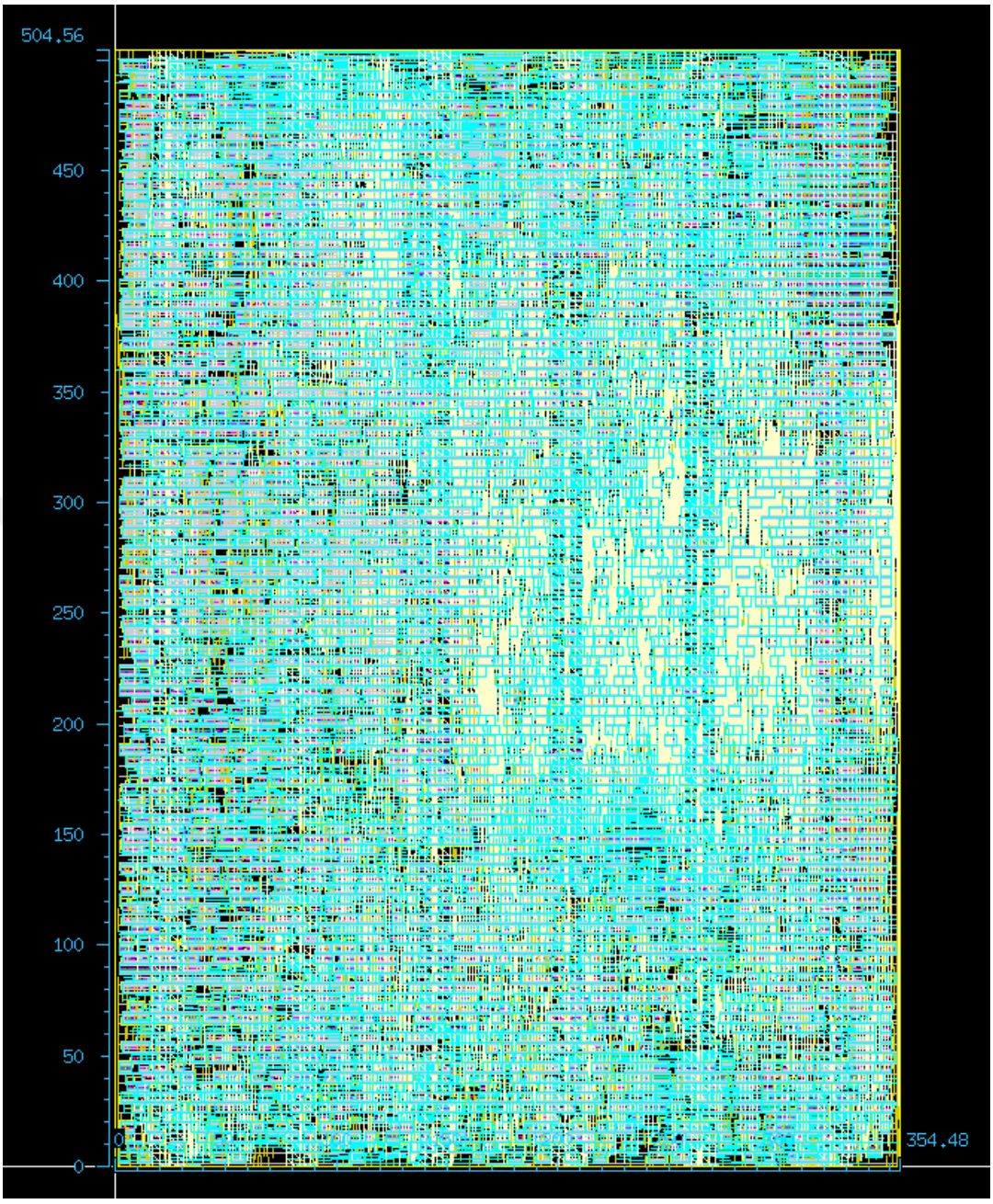


Figure 3.20. Final layout view of the controller unit. The total layout area of the circuit is 354.5 µm x 504.6 µm in 0.18 µm CMOS technology.

3.5 Memory Blocks

3.5.1 Introduction

Designed ASIC has three main memory blocks: configuration memory, image memory, and calibration memory. Memory blocks are designed as SRAM. Each memory block is implemented as separate module. Apart from these memory units, a 16-bit command memory is also implemented within the serial interface. As stated in Section 3.3, first two MSBs of the address are memory selection bits. Configuration of these two bits selects one of the three memories or the command memory. Following sections give details on implementation of these three memory blocks.

3.5.2 Configuration Memory

The configuration memory holds the programmable values for several settings for ASIC flow. It holds begin and end limits, some of the enable/disable information and configuration data for analog blocks. In addition, it holds auxiliary data for the image correction operations like the locations of the dead pixels. The configuration memory is a latch type SRAM which can hold 63x16-bits. Memory outputs stores the settings for the chip in varying dimensions. Since the ASIC is designed to accept 32 bits as 16-bit address and 16-bit data, each of the configuration memory sections are designed to hold 16-bit data. Figure 3.21 shows the schematic view of a section of the configuration memory for one bit. Each section of the configuration memory consists of a 16-bit shadow memory, 16-bit actual memory, 16-bit address decoder and logic for control signals. The shadow memory serves as a cache to hold the incoming data before issuing it to the output. This is due to the content of the configuration memory able to affect the readout settings and image flow.

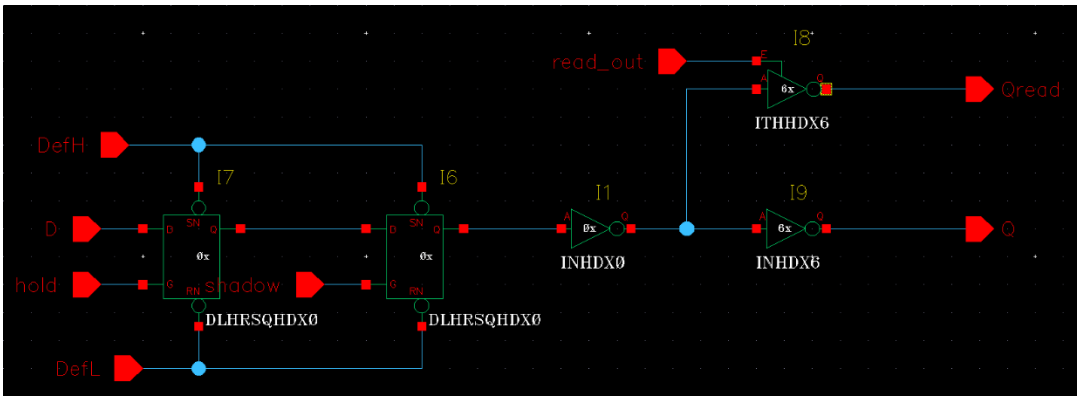


Figure 3.21. Schematic view of a section of the configuration memory for one bit.

The configuration memory is designed to be accessed only from the serial interface and read only to the controller unit. When a new single word write operation is issued by the serial interface, if first 2 bits of the address are 01, the serial interface generates a master enable signal to the configuration memory and inhibits writes and reads to other memories. 16-bit address and data bus lay along the configuration memory circuit. Particular section of the memory decodes the address and writes the data presented in the data bus to the shadow memory of that section. The content of the shadow memory is latched to actual memory with an internal hold signal. This signal is asserted immediately if the controller unit is in idle mode. However, if imaging mode of the controller unit is active, the controller unit overrides this hold signal and latches the data to actual memory when a frame read finishes.

The schematic design of the configuration memory is implemented by schematic capture method and the layout of the design is implemented by semi-automated design tools. The final layout view of the configuration memory is shown in Figure 3.22. The final layout area of the configuration memory is 288.6 µm x 570 µm. Some specifications and design utilization for the configuration memory is summarized in the Table 3.8 below.

Table 3.8. Final design specifications of the configuration memory.

Specification	Value
Pin Count	1059
Layout area	288.6 µm x 570 µm
Total Power Consumption	0.18 uW
Clock frequency	N/A
Number of logic cells	5734

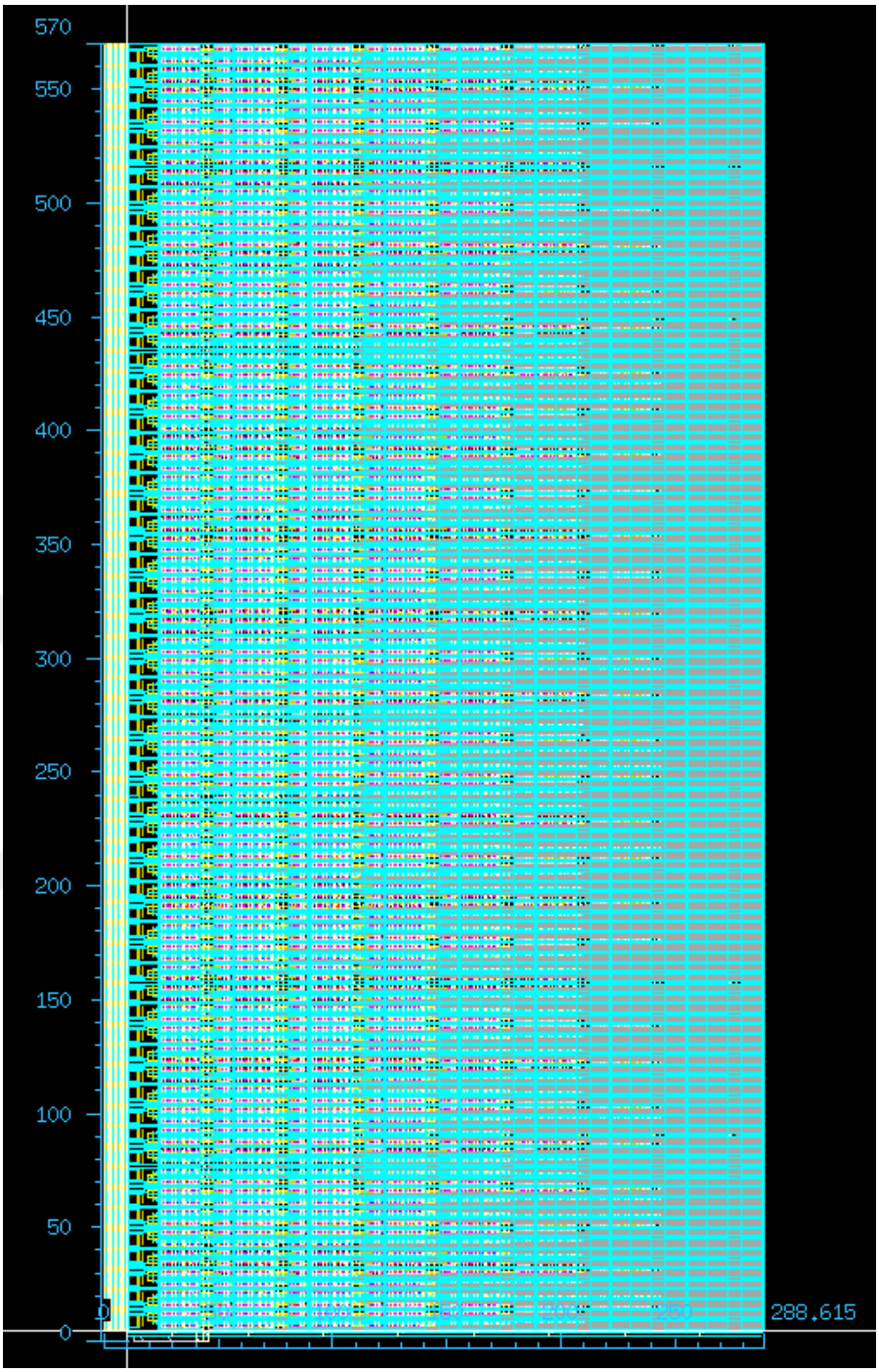


Figure 3.22. Final layout view of the configuration memory. The total layout area of the circuit is 288.6 µm x 570 µm in 0.18 µm CMOS technology.

3.5.3 Image Memory

The main function of the image memory is to store the finalized infrared image frame before sending it to external firmware. It also serves as a temporary memory for some of the image correction operations like CDS and averaging. The image memory is a flip flop type SRAM which can hold 17x16x16-bits. The image memory consists of a row decoder, a column decoder and memory core. 16x16x16-bits of the image memory is reserved for an image frame whereas the remaining 1x16x16-bits are reserved for internal measurement data. The pixels are represented by 16-bit values and therefore each of the image memory sections are designed to hold 16-bit data. Figure 3.23 shows the schematic view of a section of the image memory for one bit. Since the image memory is the main storage for incoming image, a shadow memory is not needed here.

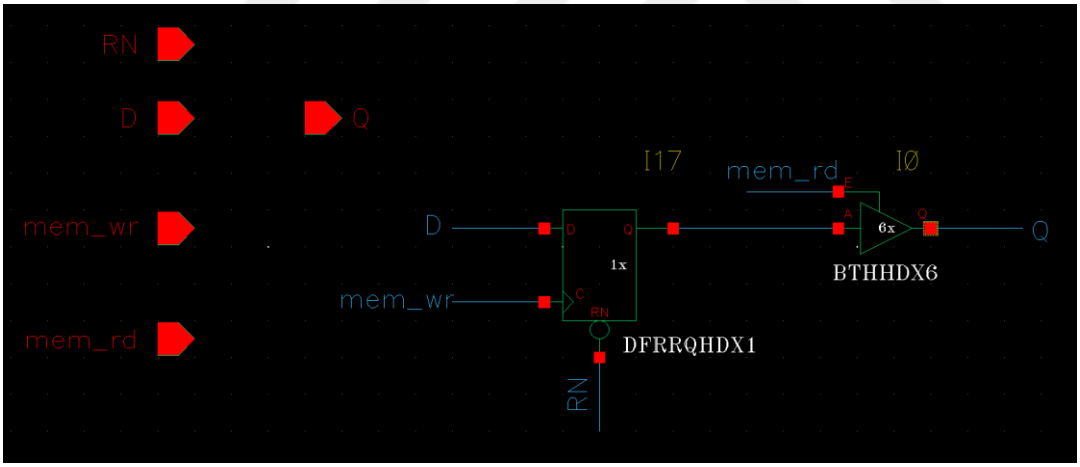


Figure 3.23. Schematic view of a section of the image memory for one bit.

The controller unit has the full access to the image memory. The controller unit perpetually stores and fetches back the pixel data in the image memory during the image correction operations. 16-bit address and 16-bit data bus is established between the image memory and the controller unit in order to provide a fast

operation. In addition, the serial interface can also access the image memory if needed. An enable bit in the configuration memory inhibits the access of the controller unit to the image memory and directs the access to the serial interface. This allows the external firmware to change the content of the image memory one by one or read it back.

The image memory schematics is designed by schematic capture tool. Blocks are organized as 4x16-bits and the signal buffers are distributed to each block. The layout of the circuit is implemented by the cutting-edge layout generation tool. After functional verifications, the designed schematic is exported to Verilog and design constraints are specified in sdc format. The layout of the circuit is generated by generation tool and static timing and power analysis are made. The final layout view of the image memory is shown in Figure 3.24. As in the controller unit final simulations and verifications are done including the annotated parasitic effects from the layout regarding the worst corners. Final layout area of the image memory is 379.7 µm x 1204.6 µm. Some specifications and design utilization for the image memory is summarized in the Table 3.9 below.

Table 3.9. Final design specifications of the image memory.

Specification	**Value**
Pin Count	76
Layout area	379.7 µm x 1024.6 µm
Total Power Consumption	1.18 mW
Clock frequency	N/A
Number of logic cells	9761

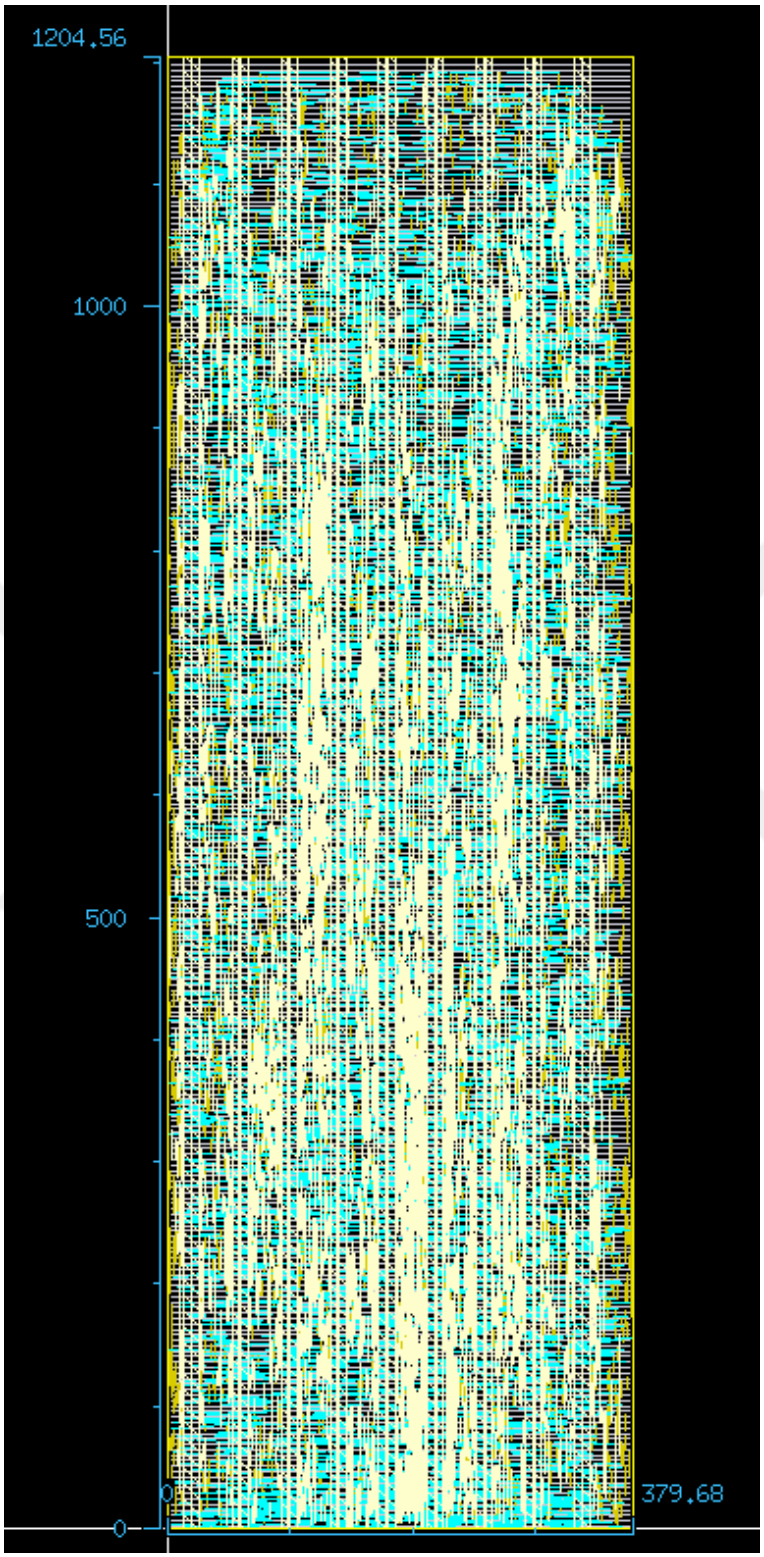


Figure 3.24. Final layout view of the image memory. The total layout area of the circuit is 379.7 µm x 1024.6 µm in 0.18 µm CMOS technology.

3.5.4 Calibration Memory

The calibration memory stores the values for the programmable capacitor array in the PFE circuit. Those values are determined by the controller unit using the automatic readout gain calibration algorithm. As an auxiliary function, the content of the calibration memory can be manually accessed and altered by the external firmware through the serial interface. The calibration memory is a flip flop type SRAM, and consists of a row decoder, a column decoder, and memory core. Figure 3.25 shows the schematic view of a section of the calibration memory for one bit. Since the automatic readout gain calibration algorithm calculates the values as 7-bits for each capacitor array, the structure of the calibration memory organized as 16x20x7-bits whose 16x16x7-bits are for actual pixels and 16x4x7-bits are for the 4 dummy column pixels.

The calibration memory circuit is indented to drive analog circuits hence it constitutes a bridge between the digital and analog domains. The circuit has some extra structures and features which differentiates it from the other memories due to this special condition. The calibration memory circuit is located among the analog blocks. This yields a long address, data and control bus between the controller unit and the calibration memory. In order to avoid timing issues due to the long routing at the top level, an extra buffer array inside the calibration memory is designed. In addition, output of the calibration memory has two separate buffer arrays one for enabling the output to analog circuits, and the other one for reading the content back through the serial interface.

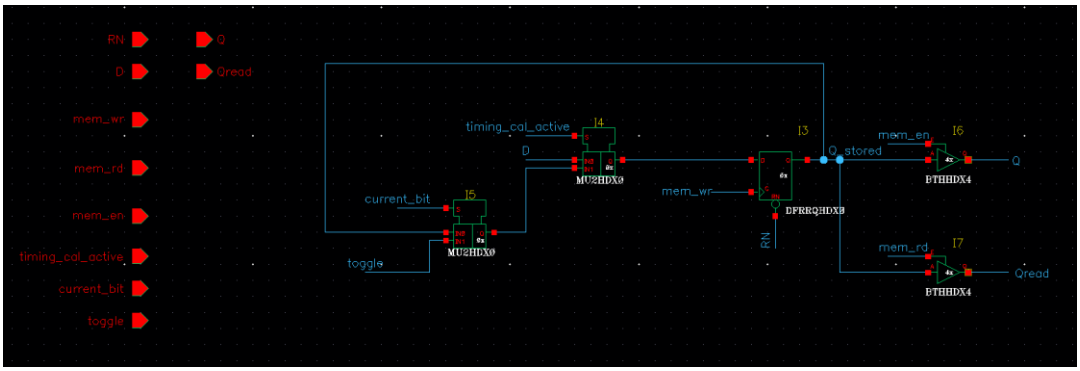
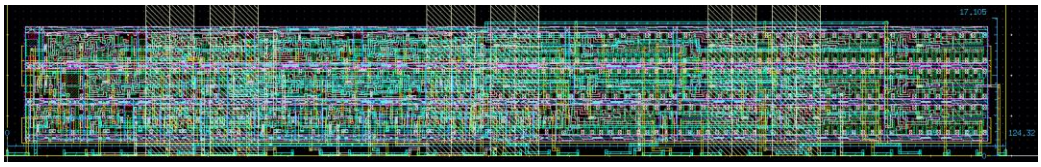


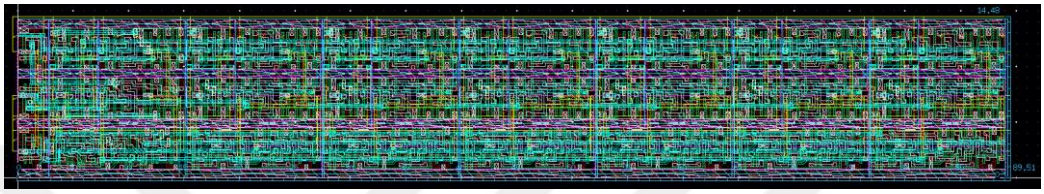
Figure 3.25. Schematic view of a section of the calibration memory for one bit.

The calibration memory contents are set to the 7-bit value 1000000 after the power up, which is the intermediate value for the capacitor array. An extra multiplexer in the input selects between the new data from the controller unit or hold the data. This structure increases the response time and lowers the data bus width for automatic readout gain calibration operation.

The 7-bit unit cell of the calibration memory is designed and verified according to RTL to GDSII flow and the layout view of this cell is shown in Figure 3.26. However, the final calibration memory utilizing this unit cell is not implemented in the final design due to the restrictions of the layout area and limitations regarding the pin locations in the top level integration stage. The calibration memory layout is located and partially distrubuted among analog blocks, so a custom design is imlemented for the calibration memory circuit. The layout view of the custom implementation of the 7-bit unit cell is shown in Figure 3.26.



(a)



(b)

Figure 3.26. Final layout view of the 7-bit unit cell of the calibration memory. (a) RTL to GDSII flow implementation, (b) custom implementation. The total layout areas of the circuits are 124.3 µm x 17.1 µm for the first circuit, and 89.5 µm x 14.5 µm for the second circuit in 0.18 µm CMOS technology.

The calibration memory is designed with schematic capture. Due to the limitations explained above, the layout of the circuit is implemented by semi-automated design tools. The final layout view of the calibration memory is shown in Figure 3.27. After the layout is completed, simulations and verifications are done including the annotated parasitic effects from the layout regarding the worst corners. The final layout area of the calibration memory is 1856.4 µm x 240.5 µm. Some specifications and design utilization for the calibration memory is given in the Table 3.10 below.

Table 3.10. Final design specifications of the calibration memory.

Specification	Value
Pin Count	230
Layout area	1856.4 µm x 240.5 µm
Total Power Consumption	0.46 uW
Clock frequency	N/A
Number of logic cells	12917

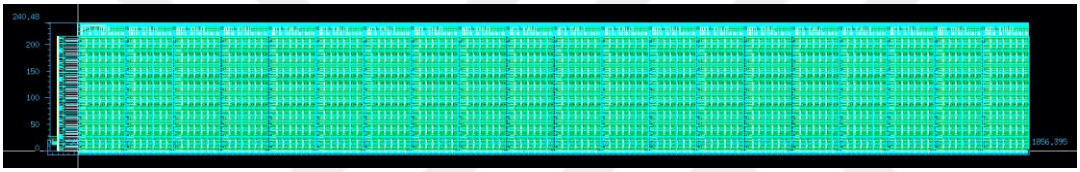


Figure 3.27. Final layout view of the calibration memory. The total layout area of the circuit is 1856.4 µm x 240.5 µm in 0.18 µm CMOS technology.

CHAPTER 4

VERIFICATION AND TEST RESULTS

4.1 Introduction

This chapter gives information about the test system development and demonstrates the verification and test results of the manufactured ASIC. Section 4.2. introduces the test setup and its constituent parts. Section 4.3. explains the firmware to drive the ASIC. Section 4.4. summarizes the verification and test results of the manufactured ASIC.

4.2 Test Setup

A test setup is designed in order to verify the ASIC. The test setup consists of a test card PCB, an FPGA card and a computer as shown in Figure 4.1. After the manufacturing and post processing steps, ASIC is placed on a ceramic LCC package and wire bonded. This ceramic package is then placed into the socket on the test card. The test card employs the FPGA board connector, an MCU with its auxiliary components, signal observation ports, power regulators to bias the ASIC, and various DACs and an external ADC to test the analog blocks. FPGA board or MCU generates digital signals to stimulate the ASIC. A computer is used to run the testing software and print the output images. In the test setup, both FPGA and MCU are utilized to test the integration of the ASIC with different system scenarios.

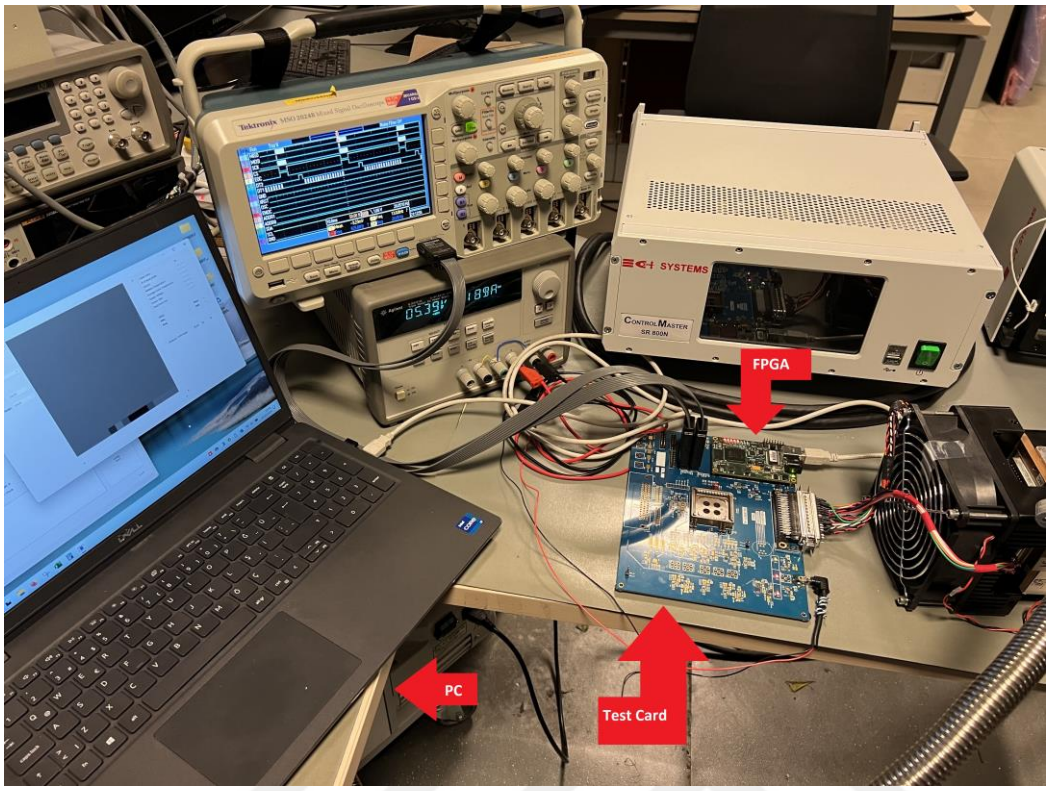


Figure 4.1. Test setup.

Interface selection depends on the device which drives the sensor, and depending on the interface selection, one of the SPI and I2C protocols can be used. Once the protocol is selected, FPGA or MCU sends the configuration data and start command to the serial interface of the ASIC. Image correction algorithms and other features can be enabled or disabled by sending proper configuration data to the ASIC. Results can be observed by an oscilloscope and on the printed image through the computer.

4.3 Firmware Development

The communication with ASIC is conducted by the serial interface. To verify the functionality of the ASIC, it should be driven by an external component like FPGA or MCU. In this regard, separate firmwares are developed for both FPGA and MCU.

FPGA card used is commercially available Opal Kelly XEM3010 board. FPGA firmware is developed in Verilog HDL, and the code is synthesized for Xilinx Spartan 3 FPGA on the board. FPGA firmware consists of three fundamental parts, the serial interface (firmware end) module, a controller module, and memory driver module. The developed firmware is responsible for sending and receiving the serial data between FPGA and ASIC, receiving infrared image, and testing data and driving the other on-board peripherals like external DACs, external ADC etc.

An MCU is used to drive the ASIC, similar to FPGA. The MCU firmware is developed in C++ language and the code is compiled and loaded into a 32-bit STM32L451 MCU. Low power features of the STM32L451 help to reduce overall system power consumption [25]. Unlike the FPGA board, MCU and its auxiliary components like oscillator, regulators, UART-to-USB bridge are placed on the test card. Like the FPGA firmware, MCU firmware is responsible for communication with the ASIC and driving other on-board peripherals.

4.4 Verification of The Functions

4.4.1 Serial Interface and Memory Verifications

The fundamental step to verify the ASIC is the serial interface verification. Since the further functions of the ASIC are enabled or disabled through the serial interface, and images are streamed through it, the serial interface should be working properly. This verification is done for three modes of operation of the serial interface of the ASIC as described in section 3.3, and both for SPI and I2C interfaces.

Firstly, single word write mode is verified by writing a word to the configuration memory. Figure 4.2 shows the results of the single word write mode verification for SPI and Figure 4.3 shows the results of the single word write mode verification for I2C. To do that, 0x0453 is written to the address 0x403F. This word changes the multiplexer selection bits in the digital test module and hence changes the digital test output signals. Outputs of the digital test signals are observed on the oscilloscope and expected waveforms are verified. Then this operation is repeated using I2C. Digital test signals DT2 and DT1 are changed in both cases so a successful write operation is verified.

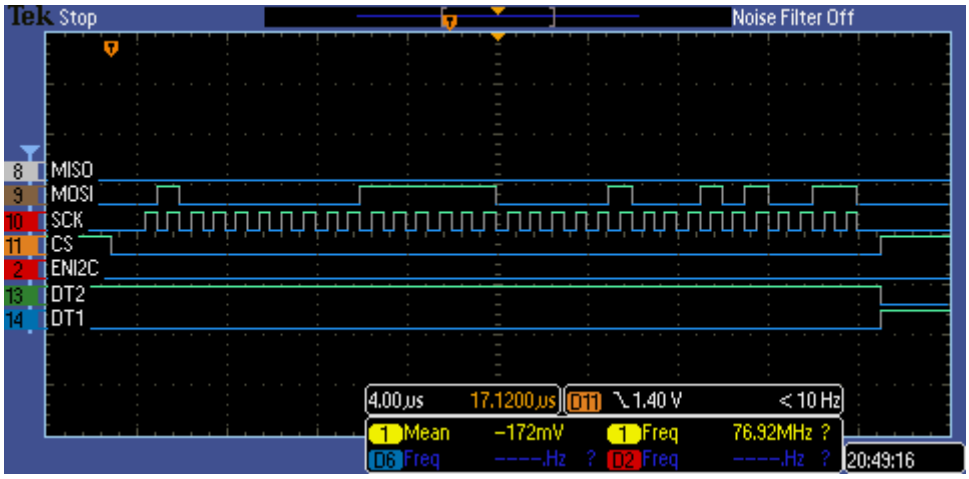


Figure 4.2. Verification results SPI single word write operation. Changing the content of the particular configuration memory address changes the states of the digital test outputs DT2 and DT1.

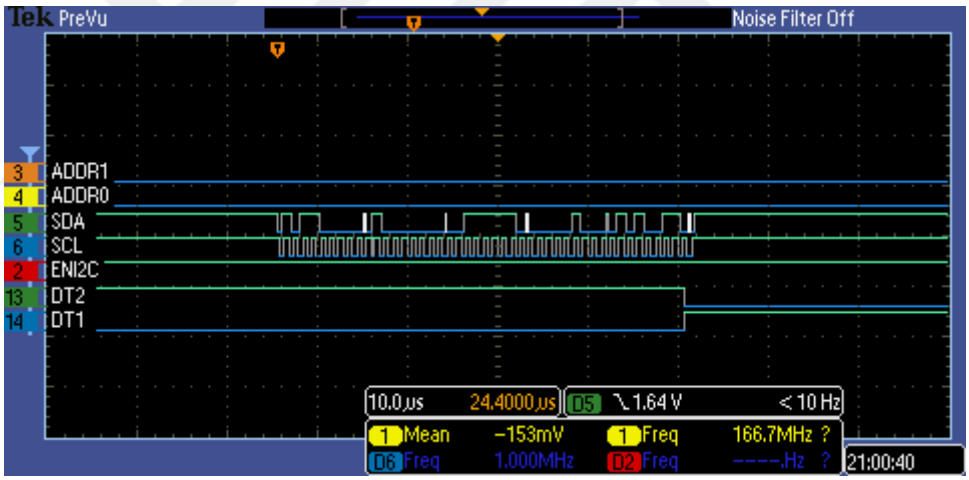


Figure 4.3. Verification results I2C single word write operation. Changing the content of the particular configuration memory address changes the states of the digital test outputs DT2 and DT1.

Secondly, the single word read mode is verified. Figure 4.4 shows the results of the single word read mode verification for SPI, and Figure 4.5 shows the results of the single word read mode verification for I2C. Content of a selected address which is 0x4033 in the configuration memory is changed to 0x01E4 and this content is read back. This operation is repeated for I2C. For the SPI verification, SPI peripheral of the ASIC should be configured to SPI read mode, on the other hand there is no such a need for the I2C verification since the I2C packages carry a dedicated write/read bit on them.

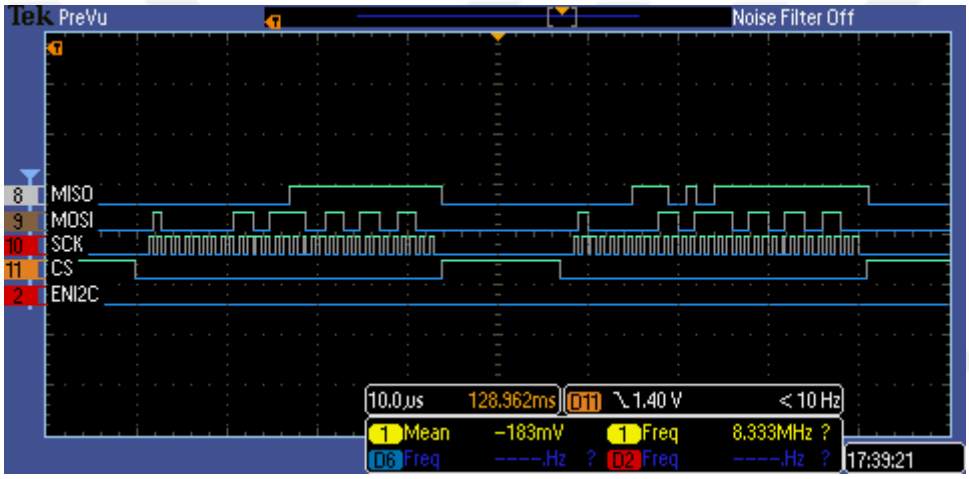


Figure 4.4. Verification results SPI single word read operation. Content of the configuration memory address 0x4033 is read back through the SPI as 0x01E4.

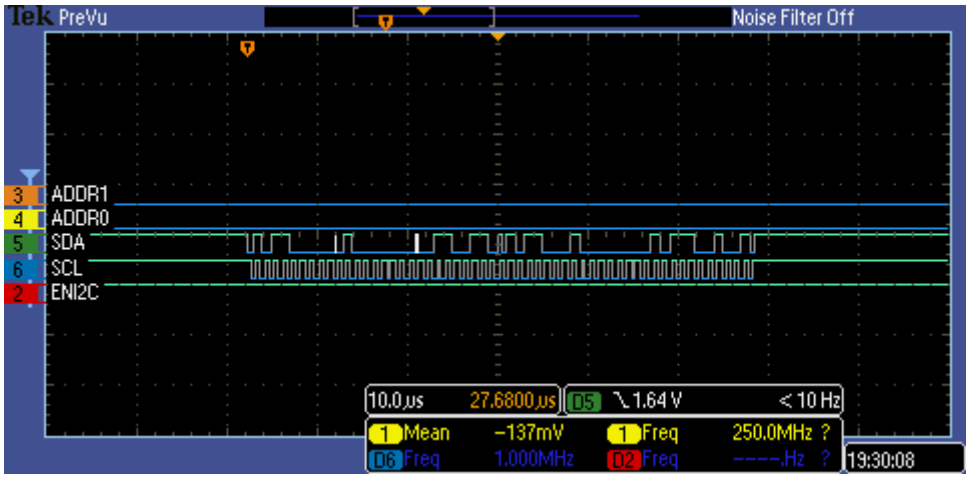


Figure 4.5. Verification results I2C single word read operation. Content of the configuration memory address 0x4033 is read back through the I2C as 0x01E4.

Memory verifications are made once the single word write and read modes are verified. Verification of the configuration memory is already done in the serial read operations above. Contents of the remaining two memories, namely the calibration memory and the image memory are changed, and these altered contents are read back through the serial interface. Since the content of the calibration memory is all 0x40, and the content of the image memory is all 0x0000 at the power-up, the write operation for these is not included. Figure 4.6 shows the calibration memory verification. In the figure 0x36 is read back from the calibration memory address 0x838F. This address corresponds to the 7th row and 15th column of the calibration memory. Figure 4.7 shows the image memory verification. In the figure 0x553E is read back from the image memory address 0xC389. This address corresponds to the 7th row and 9th column of the image memory.

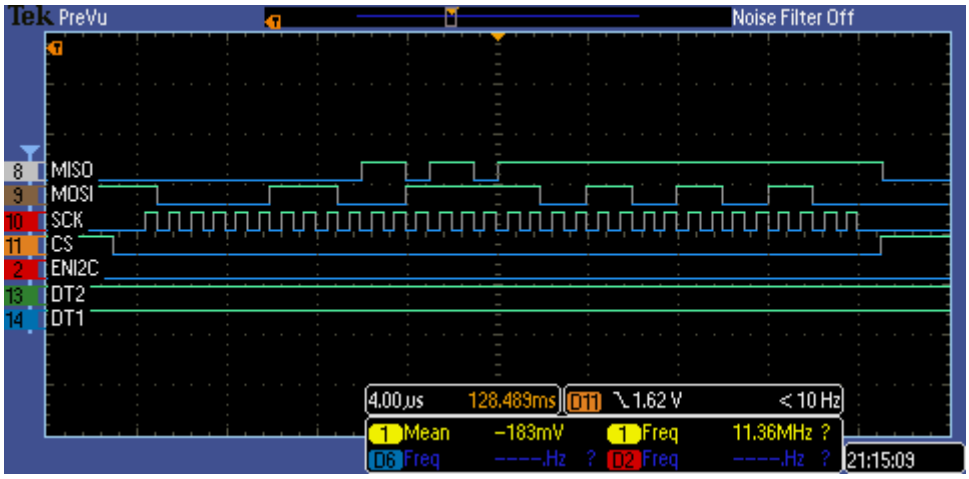


Figure 4.6. Verification results calibration memory. Content of the calibration memory address 0x838F is read back through the SPI as 0x36.

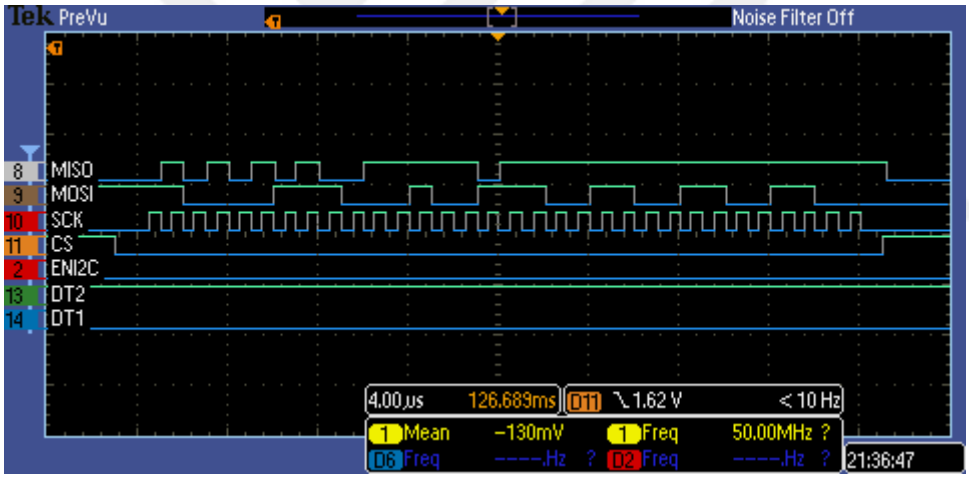


Figure 4.7. Verification results image memory. Content of the image memory address 0xC389 is read back through the SPI as 0x553E.

Lastly, the video output mode of the serial interface is verified. The oscilloscope captures of SPI and I2C waveforms related to this operation is shown in Figure 4.8 and Figure 4.9, respectively. To verify this operation, first START_IMAGING command is sent to the ASIC. This command runs the state machine of the controller unit and hence starts the image stream. Image for one frame is stored in the image memory of the ASIC. Then, end of conversion (EOC) signal is asserted by the ASIC. Firmware catches this signal and initiates a long SPI or I2C read for 1 frame.

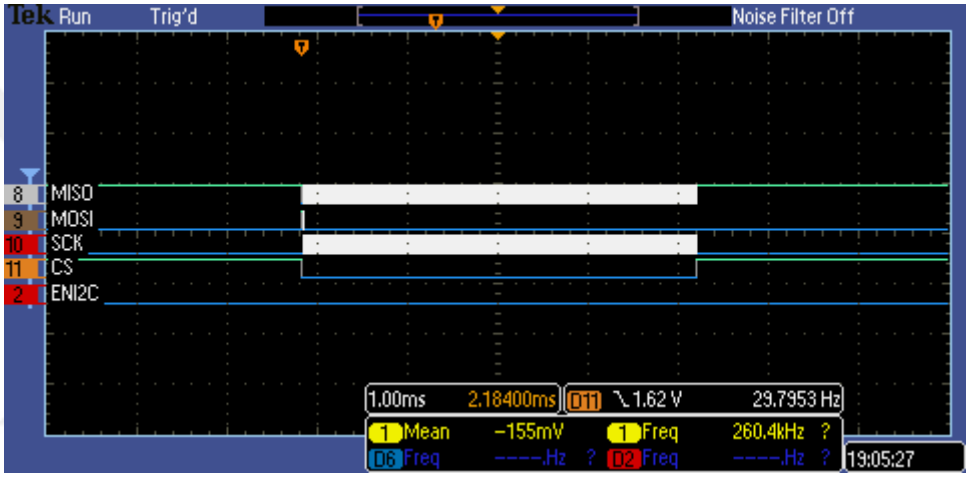


Figure 4.8. Verification results SPI video output mode. The SPI package in the figure carries an infrared video frame of 17x16x16-bit pixels.

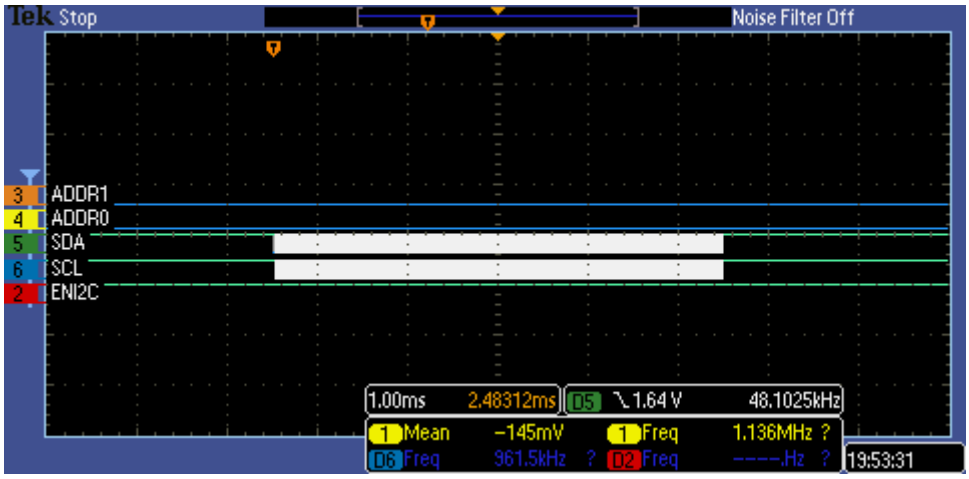


Figure 4.9. Verification results I2C video output mode. The I2C package in the figure carries an infrared video frame of 17x16x16-bit pixels.

4.4.2 Automatic Readout Gain Calibration Verification

First image correction function to be verified is automatic readout gain calibration. First, ASIC is configured to calibration mode by setting the calibration mode bits. Calibration converge value is set to 32768, and one of the digital test signals is set to *controller_busy* signal. Then, START_CALIBRATION command is sent to ASIC. After 7 consecutive frames, calibration is done and during the calibration the controller unit busy signal is asserted by the ASIC. This signal is observed in the oscilloscope. Since the automatic calibration operation is done with consecutive frames, and there is no EOC and video stream time after frames, this operation takes less than 7 *vsyncs*. Figure 4.10 shows the automatic readout gain calibration operation and Figure 4.11 shows the histograms of the images (a) before and (b) after the calibration. Standard deviation of the pixels reduced from 56 to 34 counts which means 39% improvement and mean of the pixels converged to the compare value of 35000 after the calibration.

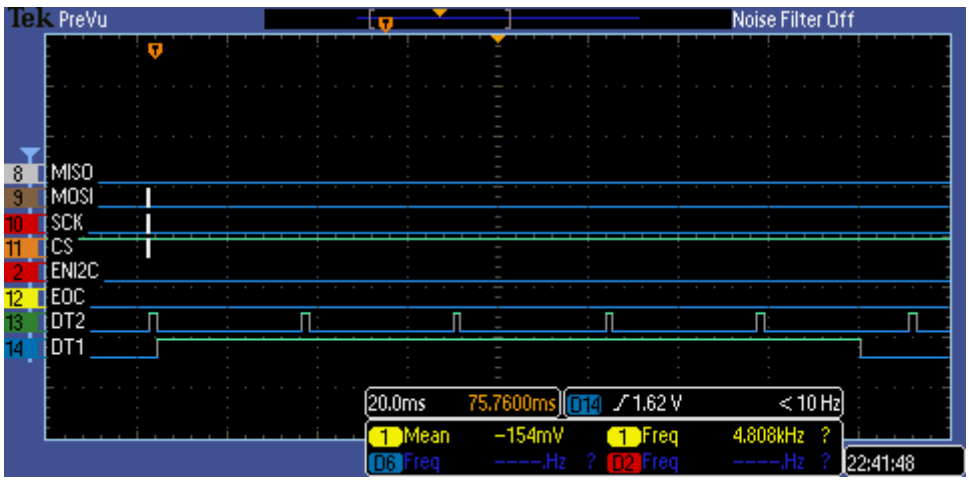
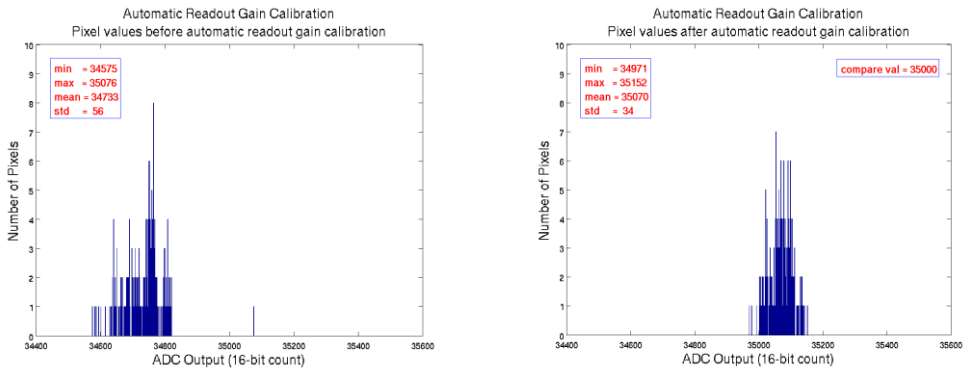


Figure 4.10. Verification results of the automatic readout gain calibration. DT1 is set to busy signal of the controller unit, and DT2 is set to vsync.

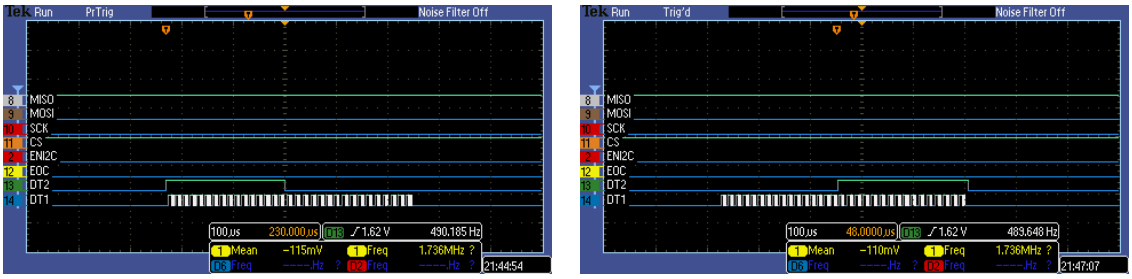


(a) (b)

Figure 4.11. Verification result of the automatic readout gain calibration. (a) Histogram of the pixels before the automatic readout gain calibration. (b) Histogram of the pixels after the automatic readout gain calibration.

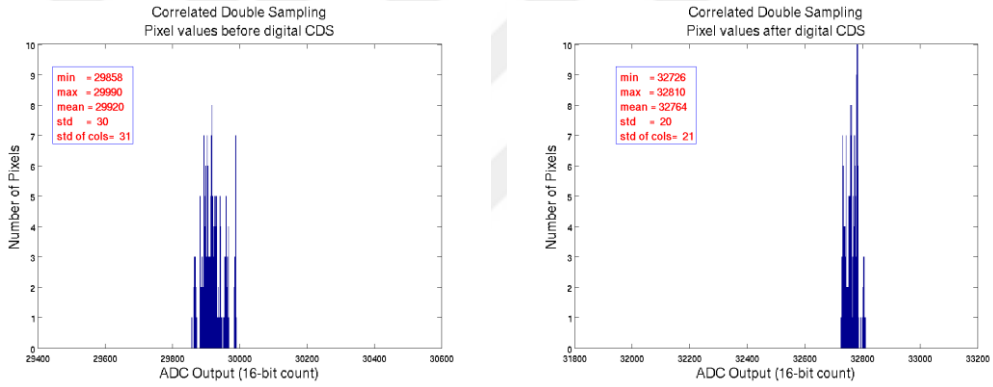
4.4.3 Correlated Double Sampling Verification

The second image correction function to be verified is correlated double sampling. RG_CDS_ENABLED bit of the configuration memory is set prior to streaming the image. In order to observe the CDS operation on oscilloscope, first digital test signal is set to CDS active signal of the controller unit, and the other digital test signal is set to ADC clock signal. This allows to observe the double scanning of a row during the CDS operation. Then START_IMAGING command is sent to the ASIC and image streaming is initiated. Figure 4.12 shows the oscilloscope waveforms of the CDS operation. DT2 test signal is set to *Vint* signal of the controller unit in the first image. DT2 test signal is set to *Vrst* signal of the controller unit in the second image. The same row is sampled twice first with *Vint* switch is open and second *Vrst* switch is open. Figure 4.13 shows the histograms of the images (a) before and (b) after the CDS. Standard deviation of the columns reduced from 30 to 20 counts which refers to 32% improvement. Offset value of 32768 is added to the after CDS image to approximate the histogram limits to the before CDS image as well as to test the internal offset function.



(a) (b)

Figure 4.12. Verification results of the CDS operation. (a) DT2 is set to *Vint* signal of the controller unit. (b) DT2 is set to *Vrst* signal of the controller unit.



(a) (b)

Figure 4.13. Verfication result of the correlated double sampling. (a) Histogram of the pixels before the correlated double sampling. (b) Histogram of the pixels after the correlated double sampling. On-chip offset is also added to the after CDS image to approximate the histogram limits to the before CDS image as well as to test the internal offset function.

4.4.4 On-Chip Averaging Verification

The on-chip averaging function is verified by the comparison of the noise levels of images with and without averaging. The ASIC is configured to averaging mode by setting the RG_EN_AVERAGING bit of the configuration memory in order to verify the on-chip averaging. For 30 frame averaging, frame rate is set to 1 fps by configuring the frame rate setting of the configuration memory. Figure 4.14 shows the oscilloscope waveforms of the averaging operation. Digital test signals are set to *vsync* signal of the controller unit and ADC clock signal to observe the ongoing averaging operation. Similar to other verifications, after required settings are done, averaging operation starts with the START_IMAGING command. To verify the operation characteristically, pixel data for 100 frames before and after the averaging is collected and standard deviation values are calculated. Figure 4.15 shows the values of all 256 pixels along 100 frames before and after 30 stage averaging operation and Table 4.1 summarizes the peak standard deviation improvement of the pixel values.

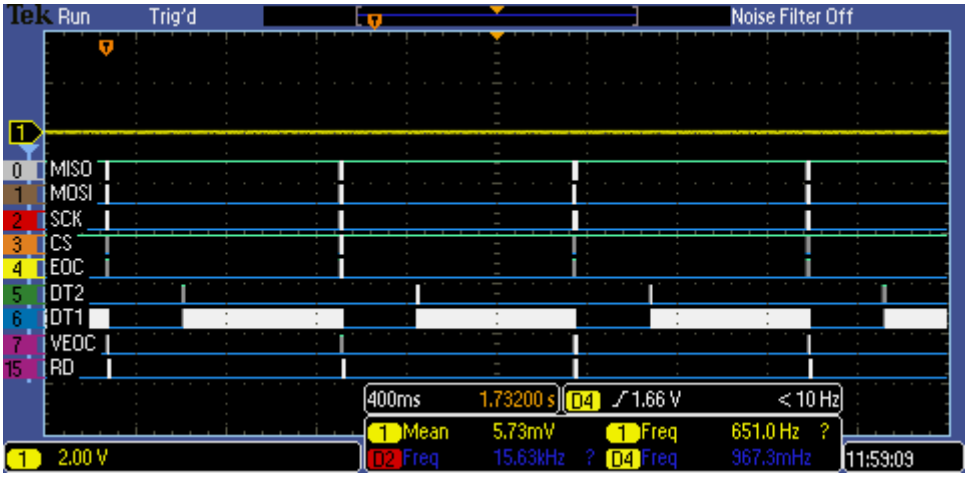


Figure 4.14. Verification results of the on-chip averaging operation. DT1 is set to adc clock signal of the controller unit, and DT2 is set to *vsync* signal of the controller unit.

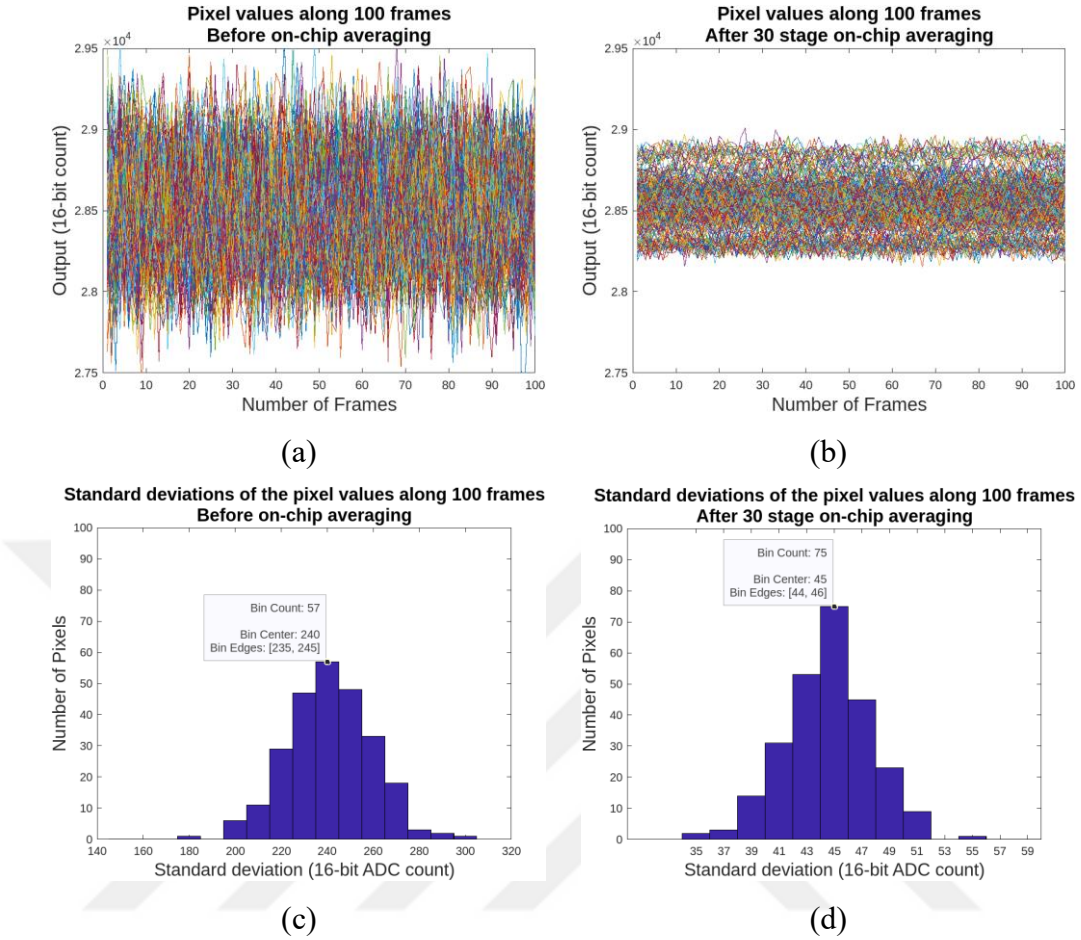


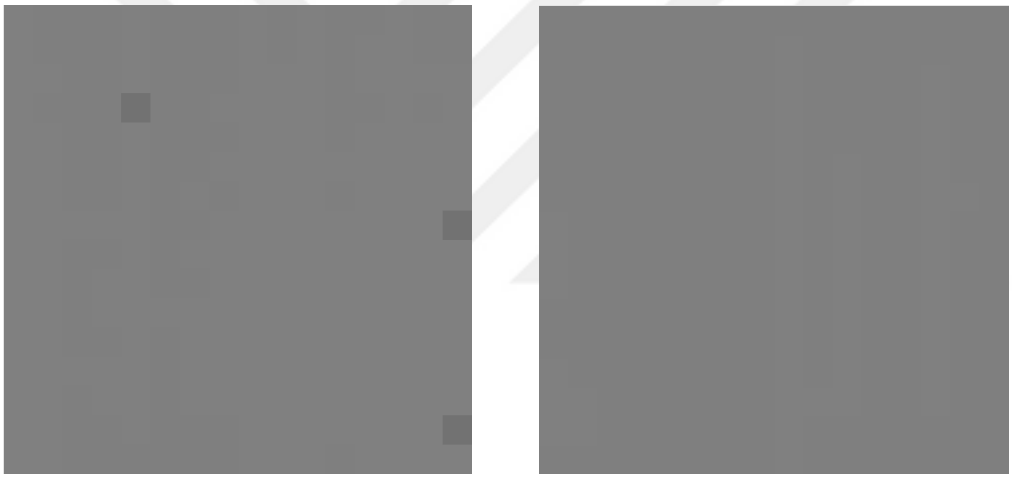
Figure 4.15. Pixel values along 100 frames (a) before and (b) after 30 stage on-chip averaging. Histograms of the standard deviations of the pixels along 100 frames (c) before and (d) after 30 stage on-chip averaging.

Table 4.1. Peak standard deviation values (16-bit ADC count) of the pixels along 100 frames before and after 30 stage on-chip averaging.

Standard Deviation	**Before on-chip averaging**	**After on-chip averaging**
Peak standard deviaion	240	45

4.4.5 Dead Pixel Correction Verification

Dead pixels are very recognizable among a 16 by 16 focal plane array so the verification of the dead pixel correction function can be done by observing the input and output images. First, RG_EN_DEAD_PIX_COR bit is set in the configuration memory. After starting the ASIC, output images are collected and observed. In order to verify the function, a dedicated sensor is prepared with 3 dead pixels. One of the dead pixels is on the corner, the other one is among the FPA, and the last one is on the edge of the FPA. Figure 4.16 shows the images before and after the dead pixel correction operation.



(a) (b)

Figure 4.16. Verification result of the dead pixel correction. (a) Image before the dead pixel correction. (b) Image after the dead pixel correction.

Finally, some images captured with the ASIC and accompanying 16x16 CMOS-IR imager are shown in the Figure 4.17. Images are captured at a distance of 2.5 meters with f1.0 LWIR lens. Pseudo coloring is added virtually to the top-right image in the software.

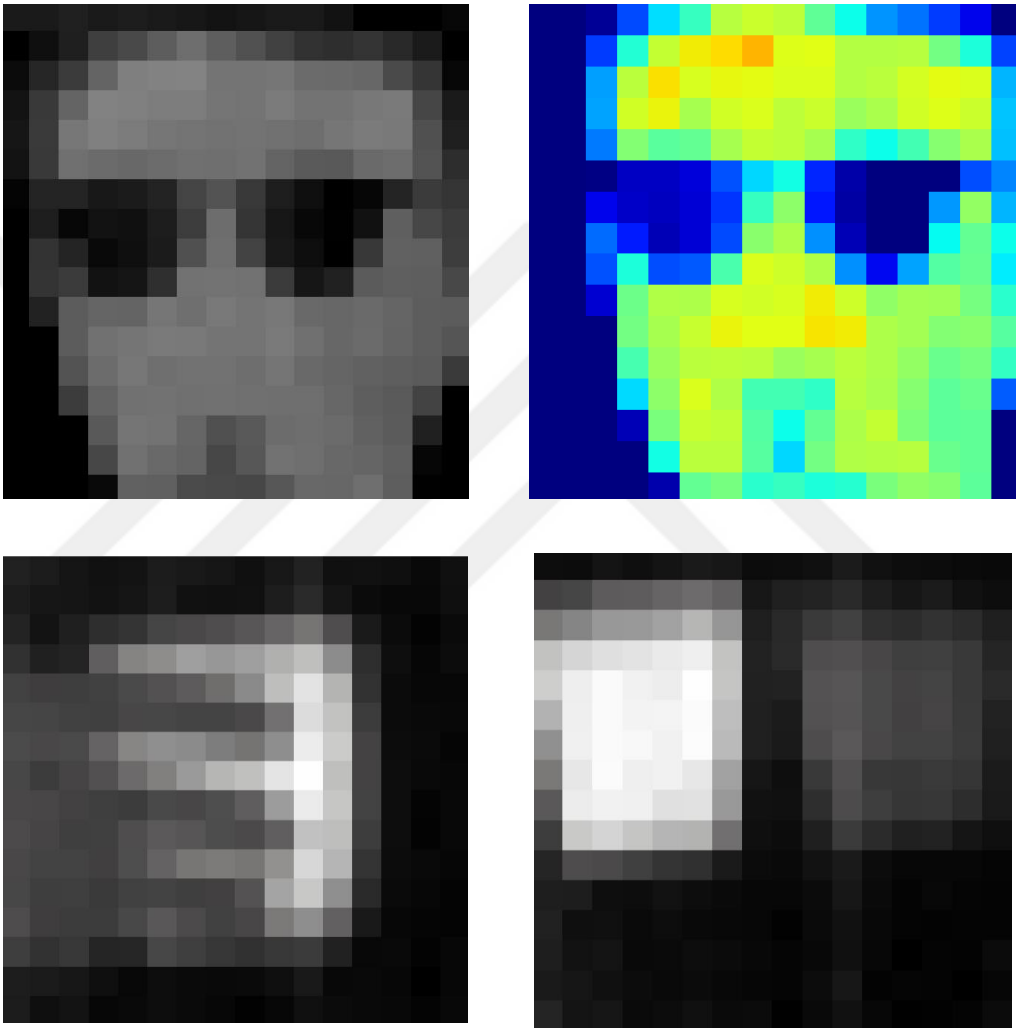


Figure 4.17. Some pictures captured using the ASIC and accompanying 16x16 CMOS-IR imager. Upper two images show a human face. Lower left image shows fingers grasping a hot cup, and lower right image shows a hot cup and a cold cup together.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CMOS infrared imagers come up with a promising solution to the cost problem of the infrared imaging systems, yet their output images still need further correction. These corrections are generally made outside of the chip. Except from some partial studies, there is not a complete on-chip solution which addresses all four image artifacts explained in this work. The main objective of this thesis is to develop an ASIC which brings an integrated solution to the fundamental image artifacts observed in CMOS infrared imagers. The ASIC is designed and implemented in 0.18 µm CMOS technology. All the functions of the design tested and verified on the manufactured silicon. The summary of the work performed in this thesis is listed below:

1. The basics of infrared imaging are introduced, and main components of a complete infrared imaging system are explained.
2. Fundamental image artifacts observed in CMOS infrared imagers are stated and possible sources of these artifacts are investigated.
3. System requirements of the ASIC are decided for image correction operations as well as for image generation requirements and communication interface development.
4. Various readout calibration methods to eliminate signal non-uniformities are researched, and novel algorithms for two most promising solutions are developed. First algorithm calibrates the gain of the integrator stage, and the second algorithm calibrates the bias voltages. These algorithms are simulated on MATLAB environment and results are compared. First algorithm is selected to be implemented on the ASIC due to its advantages on layout area and design effort.

5. Correlated double sampling methods are investigated and possibility of a digital implementation is evaluated. Presence of an image memory module to hold the reset values of the pixels makes the digital implementation possible. Therefore, a digital correlated double sampling algorithm is developed to run on the ASIC.
6. A practical algorithm to implement on-chip frame averaging is developed and the concept is proved on MATLAB environment. In the proposed approach the algorithm uses the results of the previous averaging stages to calculate the averaging result hence it promises smaller memory area, less logic utilization and fewer clock cycles than the classical averaging approach.
7. An on-chip dead pixel correction method is developed. In this method, 3 possible dead pixel regions are determined, and different correction algorithms are applied for these different regions.
8. A flexible serial interface module which runs at 1MHz serial clock is designed and implemented. A word size for the serial interface is decided to be 16-bits, 8-bits for address and 8-bits for data. In the serial interface both SPI and I2C peripheral circuits are implemented. SPI peripheral circuit is designed to operate as SPI mode 0 (CPOL:0, CPHA:0). I2C peripheral circuit is designed to meet the requirements for I2C fast mode plus (FM+). In addition, video output mode like video over SPI and video over I2C is included in this module.
9. A versatile controller unit is designed and implemented according to RTL to GDSII flow. The design is optimized to work at 5 MHz clock speed. The controller module drives the sensor and analog blocks to stream the image and implements the image correction architectures. The controller unit contains the logic for automatic readout gain calibration, correlated double sampling, frame averaging and dead pixel correction algorithms.
10. Three different SRAM memory blocks are designed and implemented. The configuration memory holds the settings for the ASIC, image memory holds the image data and supports the image correction operations, and the

calibration memory holds the values for the automatic readout gain calibration results.

11. A test setup including the test PCB is prepared to verify all functions of the ASIC. FPGA and MCU firmwares are developed. All functions of the ASIC are tested and verified on the manufactured silicon. With the automatic readout gain calibration the standard deviation of the pixel signals are reduced by 39%. With the correlated double sampling implementation, the readout offsets are removed and the standard deviation among the columns are reduced by 32%. the high frequency noises are successfully suppressed with the on-chip averaging and the signal noise is reduced by 82%. The dead pixels are properly eliminated with the dead pixel correction.

On-chip implementation of the image correction algorithms is a remarkable step forward to improve the image quality of the CMOS infrared imagers in an all-in-one solution. However, some additional features can be added to the design in order to further improve the functionality of the proposed ASIC. Future work regarding these improvements is listed below:

1. Output data of the ASIC is formatted as 16-bit unsigned counts which are relative to incoming infrared light intensity. An algorithm which converts this raw data to absolute temperature values can be developed within the ASIC. Since the sensitivities of pixels vary among different sensors, particular calibration data for this algorithm should be stored in each individual chip. This operation requires a non-volatile memory structure like an EEPROM. Therefore, an EEPROM should be integrated to the ASIC design in the future versions of the chip.
2. The design of the ASIC is optimized for 5 MHz clock speed since the accompanying sensor has 16x16 pixels in the FPA which do not require higher clock speeds. However, FPA sizes of the CMOS infrared imagers nowadays can vary up to 640x480 pixels. In order to catch up with such larger FPA size requirements, clock speed of the ASIC should be increased. In addition, the serial interface and the controller unit works in different clock

domains. In video output mode, a fast data transfer happens between these blocks. In higher clock rates, this may yield synchronization problems. Therefore, in order to obtain higher clock speeds safely, a PLL design should be included in the next ASIC architecture.

REFERENCES

[1] A. Rogalski, “Infrared detectors: an overview,” *Infrared Phys Technol*, vol. 43, no. 3–5, pp. 187–210, Jun. 2002, doi: 10.1016/S1350-4495(02)00140-8.

[2] Photonics Spectra, “SWIR Applications and Challenges: A Primer.” Accessed: Mar. 27, 2024. [Online]. Available: https://www.photonics.com/Articles/SWIR_Applications_and_Challenges_A_Primer/a56646

[3] K. Qian, “Variability Modeling and Statistical Parameter Extraction for CMOS Devices,” Berkeley, Jun. 2015. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-165.html

[4] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, “Matching properties of MOS transistors,” *IEEE J Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct. 1989, doi: 10.1109/JSSC.1989.572629.

[5] A. Sheikholeslami, “Process variation and pelgrom’s law [circuit intuitions],” *IEEE Solid-State Circuits Magazine*, vol. 7, no. 1, pp. 8–9, Dec. 2015, doi: 10.1109/MSSC.2014.2369331.

[6] P. A. Coelho, J. E. Tapia, F. Pérez, S. N. Torres, and C. Saavedra, “Infrared light field imaging system free of fixed-pattern noise,” *Sci Rep*, vol. 7, no. 1, p. 13040, Oct. 2017, doi: 10.1038/s41598-017-13595-7.

[7] M. Tepegoz and T. Akin, “Resistance non-uniformity correction method using bias heating for resistive type uncooled microbolometer FPAs,” B. F. Andresen, G. F. Fulop, and P. R. Norton, Eds., Apr. 2007, p. 65423T. doi: 10.1117/12.726570.

[8] M. H. Çöloğlu, “Advanced Circuit Architectures for Readout Electronics of Low-Cost Microbolometer Focal Plane Arrays,” M.Sc. Thesis, METU, Ankara, 2017.

[9] Quilligan et al., “Thermopile Bias Method for Low Voltage Infrared Readout Integrated Circuits,” US 10,746,594 B1, Aug. 18, 2020

[10] M. Kim, S. Park, K. Lee, and H.-J. Yoo, “Uncooled Infrared Micro-Bolometer FPA for Multiple Digital Correlated Double Sampling,” *IEEE Photonics Technology Letters*, vol. 30, no. 6, pp. 517–520, Mar. 2018, doi: 10.1109/LPT.2018.2792491.

[11] A. Moini, *Vision Chips*. Boston, MA: Springer US, 2000. doi: 10.1007/978-1-4615-5267-3.

[12] Gao Jun, Chen Zhongjian, Lu Wengao, Cui Wentao, and Ji Lijiu, “Correlated double sample design for CMOS image readout IC,” in *Proceedings. 7th International Conference on Solid-State and Integrated Circuits Technology, 2004.*, IEEE, 2004, pp. 1437–1440. doi: 10.1109/ICSICT.2004.1436861.

[13] M. Zou, J. Zhang, S. Zhong, Z. Li, and L. Yao, “Low-light-level CMOS imaging sensor with CTIA and digital correlated double sampling,” *Analog Integr Circuits Signal Process*, vol. 101, no. 3, pp. 449–461, Dec. 2019, doi: 10.1007/s10470-019-01533-8.

[14] M. F. Snoeij, A. J. P. Theuwissen, K. A. A. Makinwa, and J. H. Huijsing, “A CMOS Imager With Column-Level ADC Using Dynamic Column Fixed-Pattern Noise Reduction,” *IEEE J Solid-State Circuits*, vol. 41, no. 12, pp. 3007–3015, Dec. 2006, doi: 10.1109/JSSC.2006.884866.

[15] Woodward Yang, Oh-Bong Kwon, Ju-Il Lee, Gyu-Tae Hwang, and Suk-Joong Lee, “An integrated 800×600 CMOS imaging system,” in *1999 IEEE International Solid-State Circuits Conference. Digest of*

Technical Papers. ISSCC. First Edition (Cat. No.99CH36278), IEEE, 1999, pp. 304–305. doi: 10.1109/ISSCC.1999.759261.

[16] D. Kim, J. Bae, and M. Song, "A High Speed CMOS Image Sensor with a Novel Digital Correlated Double Sampling and a Differential Difference Amplifier," *Sensors*, vol. 15, no. 3, pp. 5081–5095, Mar. 2015, doi: 10.3390/s150305081.

[17] S. Kawahito, N. Kawai, and Y. Tadokoro, "A High-Sensitivity Oversampling Digital Signal Detection Technique for CMOS Image Sensors Using Non-destructive Intermediate High-Speed Readout Mode," Hamamatsu, 2008.

[18] W. Kester, *Practical Design Techniques for Sensor Signal Conditioning*, vol. 1. Analog Devices, 1999.

[19] H. Hoffmann, "Unsupervised Learning of Visuomotor Associations," Ph.D. Thesis, Bielefeld University, Bielefeld, 2005.

[20] K. Das, D. Nath, and S. N. Pradhan, "FPGA and ASIC realisation of EMD algorithm for real-time signal processing," *IET Circuits, Devices & Systems*, vol. 14, no. 6, pp. 741–749, Sep. 2020, doi: 10.1049/iet-cds.2019.0322.

[21] R. Celestre, M. Rosenberger, and G. Notni, "A novel algorithm for bad pixel detection and correction to improve quality and stability of geometric measurements," *J Phys Conf Ser*, vol. 772, p. 012002, Nov. 2016, doi: 10.1088/1742-6596/772/1/012002.

[22] A. E. Mudau, C. J. Willers, D. Griffith, and F. P. J. le Roux, "Non-uniformity correction and bad pixel replacement on LWIR and MWIR images," in *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, IEEE, Apr. 2011, pp. 1–5. doi: 10.1109/SIECPC.2011.5876937.

[23] B. M. López-Portilla, W. Valenzuela, P. Zarkesh-Ha, and M. Figueroa, “A CMOS Image Readout Circuit with On-Chip Defective Pixel Detection and Correction,” *Sensors*, vol. 23, no. 2, p. 934, Jan. 2023, doi: 10.3390/s23020934.

[24] NXP Semiconductors, “UM10204 I 2 C-bus specification and user manual,” Oct. 2021.

[25] ST Microelectronics, “STM32L451xx Ultra-low-power Arm ® Cortex ®-M4 32-bit MCU+FPU, 100DMIPS, up to 512KB Flash, 160KB SRAM, analog, audio Datasheet,” Oct. 2020. [Online]. Available: www.st.com