Advanced sensing and processing methodologies for ISFET based DNA sequencing

by

Yuanqi Hu

March 2015

A Report submitted for the degree of Doctor of Philosophy of Imperial College London

Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Declaration of Originality

I hereby declare that this thesis and the work described is my own and that everything else is appropriately referenced and attributed.
Declaration of Copyright

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.
Abstract

The growing need to obtain large amounts of genomic data for various applications has motivated efforts to drop the cost and time of DNA sequencing and assembly. At the forefront of these, semiconductor based sequencing using Ion-Sensitive Field Effect Transistors (ISFETs) shows great promise. This thesis explores methods to improve ISFET sensor performance using novel front-end topologies in CMOS in addition to introducing new real-time parallel processing methods to allow more robust and rapid DNA sequencing and assembly in hardware.

The novel front-ends utilise capacitive feedback. By doing so, all the existing challenges in ISFET sensors such as trapped charge, sensitivity loss and drift can be solved. Three different topologies (two-stage, single-stage and 3-Transistor) are discussed and compared. The single stage front-end is also found to be the most suitable structure for implementing large arrays of sensors.

A novel automatic calibration system is designed to compensate for the gain mismatches in the sensor array, which monitors the amplitude of high frequency sine waves superimposed on the chemical signals. A trade-off between speed and resolution is resolved by adding additional lowpass filters in the loop.

Following this, a full system comprising a $32 \times 32$ ISFET array, an automatic gain calibration system, control logics and SPI is implemented. The correlated double sampling system which eliminates the offset problem is realized in digital domain. An SPI protocol is used to send off the digitalised data to off-chip memory, as well as the data retrieval. The test results indicate a good performance in offset cancellation and gain consistency.

Finally a real-time DNA fragment comparison system is implemented in the FPGA for DNA assembly. To handle with the incomplete data set during the sequencing time, a novel hybrid comparison algorithm is proposed. The original all-against-all comparison step in the OLC method is firstly decomposed into successive window-against-window comparison phases, and then dynamic programming is grafted on the exact comparison to achieve high speed but error-tolerant computation. Hierarchical implementation in FPGA is represented where processing units are paralleled and controlled by one global controller. The validation
of the proposed system is proven by the assembly of three real DNA sets even with deliberate errors introduced.
Contents

1 Introduction .................................................. 3
   1.1 Motivation ................................................... 3
   1.2 Research Objectives ....................................... 6
   1.3 Thesis Organisation ...................................... 7
      1.3.1 DNA sequencing using semiconductors ............... 7
      1.3.2 ISFET interface based on capacitive feedback ........ 8
      1.3.3 Automatic Gain Control Systems for ISFET array compensation . . . . 8
      1.3.4 An ISFET Chemical Sensing Array with Automatic Gain Calibration ... 8
      1.3.5 Real-Time Sequence Comparison for DNA Assembly .............. 9

2 DNA Sequencing methods and the semiconductor based approach ...... 12
   2.1 Introduction .................................................. 12
   2.2 DNA properties and base pairing ................................ 13
   2.3 DNA incorporation, hybridisation and sequencing ................. 14
   2.4 Next generation DNA sequencing ................................... 15
      2.4.1 454 Pyrosequencing ..................................... 16
      2.4.2 Illumina/Solexa system ................................... 17
      2.4.3 AB SOLiD .................................................. 18
## CONTENTS

2.4.4 Ion Personal Genome Machine ........................................ 19
2.4.5 Summary of different sequencing technologies ............................ 21
2.5 Semiconductor based chemical sensors ................................... 22
2.6 Ion Sensitive Field Effect Transistor .................................... 24
2.7 ISFET sensing mechanism ................................................. 24
   2.7.1 Electrolyte-Insulator Interface ...................................... 25
   2.7.2 Helmholtz & Gouy-Chapman model .................................... 26
2.8 Extended-gate ISFET & Practical Modelling ................................. 28
2.9 ISFET Non-ideal effects .................................................. 31
2.10 ISFET front-end circuits ............................................... 32
   2.10.1 Classical front-ends ................................................ 32
   2.10.2 Current mode techniques ........................................... 33
   2.10.3 Programmable gate techniques ..................................... 33
   2.10.4 pH-to-time techniques .............................................. 34
   2.10.5 Differential readout ............................................... 35
2.11 Desirable features of ISFET front-end designs ............................ 35

3 **ISFET interfaces based on capacitive feedback** .......................... 44
    3.1 Introduction ........................................................ 44
    3.2 A novel ISFET readout circuit ........................................ 45
       3.2.1 System Analysis .................................................. 45
       3.2.2 Simulated System Operation ...................................... 47
       3.2.3 Stability analysis ............................................... 48
       3.2.4 Noise analysis .................................................. 50
3.2.5  Leakage analysis ........................................ 50
3.2.6  Design tradeoff when choosing capacitors ............... 53
3.3  Design and Fabrication ................................... 54
3.4  Test setup .............................................. 55
3.5  Measured Results ....................................... 56
  3.5.1  Test of immunity to capacitive division .................. 56
  3.5.2  Test of pH sensitivity ................................ 59
  3.5.3  Test of long-term drift .............................. 60
  3.5.4  Noise measurement ................................... 62
  3.5.5  Chemical reaction monitoring .......................... 62
3.6  Front-end Summary ...................................... 64
3.7  Single stage capacitive feedback interface ................... 65
  3.7.1  Pixel Schematic ..................................... 66
  3.7.2  Design and Fabrication ............................... 66
  3.7.3  Test of Amplification ................................ 68
  3.7.4  Test of pH sensitivity ............................... 68
  3.7.5  Test of Noise ....................................... 69
  3.7.6  Summary of single stage interface ..................... 69
3.8  Three-transistor ISFET readout circuit ....................... 70
  3.8.1  Pixel Schematic ..................................... 71
  3.8.2  Design and Fabrication ............................... 71
  3.8.3  Test of Amplification ............................... 72
  3.8.4  Test of chemical response ............................ 73
  3.8.5  Test of noise ....................................... 73
6 Real-Time Sequence Comparison for DNA Assembly

6.1 Introduction...........................................130

6.2 Existing assembly algorithms............................131

6.2.1 Greedy.............................................131

6.2.2 Overlap-layout-consensus............................132

6.2.3 De Bruijn Graph.....................................133
CONTENTS

7.2.1 Optimisation and development in front-ends .................. 163
7.2.2 Potential in calibration system .......................... 164
7.2.3 Fully integrated DNA sequencing & assembly system .......... 164
7.3 Personal perspective ........................................ 164

A Publications arising from this work ................................ 166
B Fabricated chips .............................................. 167
C DNA Detection in CMOS ....................................... 170
D Codes of Fragments Comparison algorithm for FPGA ............. 174
E Codes of Control logic in the Sensing system ..................... 204
F PCB test circuits .............................................. 210
G pH buffer Preparation ........................................ 213
## List of Figures

1.1 Typical flow of DNA sequencing ........................................ 5
1.2 Flow for the proposed hardware based DNA sequencing system .... 7

2.1 Double-Helix model of DNA and its chemical structure ............... 13
2.2 Biochemical mechanism of Nucleotide incorporation which release hydrogen ions and phosphate groups .................................. 14
2.3 DNA bonded beads immobilised in the well ................................ 16
2.4 Demonstration of Pyrosequencing process .................................. 17
2.5 Demonstration of sequencing process for Solexa system ................. 18
2.6 Demonstration of sequencing process for SOLiD system .................. 19
2.7 Detection Diagram of completing 5 rounds .................................. 20
2.8 1-nucleotide incorporation signal [30] ..................................... 20
2.9 The MOSFET and the ISFET .................................................. 25
2.10 Site binding on the surface between the insulator and solution .......... 26
2.11 Illustration showing (a) SiO$_2$-electrolyte interface in a ISFET, (b) the ionic distribution forming the Helmholtz and Gouy-Chapman layers, (c) associated potential changes and (d) equivalent capacitor .......................... 27
2.12 Illustration of an ISFET in an unmodified CMOS technology and its macro-model .................................................. 29
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.13</td>
<td>Non-ideal effects present in CMOS based ISFETs [74]</td>
<td>31</td>
</tr>
<tr>
<td>2.14</td>
<td>The schematic of a constant voltage constant current front-end</td>
<td>32</td>
</tr>
<tr>
<td>2.15</td>
<td>The schematic of a current mode ISFET front-end</td>
<td>33</td>
</tr>
<tr>
<td>2.16</td>
<td>The schematic of programmable gate ISFET front-end</td>
<td>33</td>
</tr>
<tr>
<td>2.17</td>
<td>The schematic of the pH-to-time front-end</td>
<td>34</td>
</tr>
<tr>
<td>2.18</td>
<td>The schematic of ISFET based Gilbert Cell</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Non-ideal issues affecting the detected signal</td>
<td>45</td>
</tr>
<tr>
<td>3.2</td>
<td>Schematic of the capacitive feedback ISFET interface</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>Transient response of the system in two different phases</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Open-loop model for stability analysis</td>
<td>49</td>
</tr>
<tr>
<td>3.5</td>
<td>Schematic for analysing noise equivalence</td>
<td>50</td>
</tr>
<tr>
<td>3.6</td>
<td>Low-leakage low-charge-injection switch</td>
<td>51</td>
</tr>
<tr>
<td>3.7</td>
<td>Current leakage under light for two different gate capacitors</td>
<td>52</td>
</tr>
<tr>
<td>3.8</td>
<td>Tradeoff in choosing the value of feedback capacitor</td>
<td>53</td>
</tr>
<tr>
<td>3.9</td>
<td>Microphotograph of fabricated test devices</td>
<td>54</td>
</tr>
<tr>
<td>3.10</td>
<td>Experimental Setup</td>
<td>56</td>
</tr>
<tr>
<td>3.11</td>
<td>Illustration of an ISFET in an unmodified CMOS technology and its macro-model</td>
<td>57</td>
</tr>
<tr>
<td>3.12</td>
<td>Amplification of ISFETs for 5 different chips</td>
<td>58</td>
</tr>
<tr>
<td>3.13</td>
<td>Step output change from pH 7 for pH 5, 6, 8 and 9</td>
<td>60</td>
</tr>
<tr>
<td>3.14</td>
<td>pH sensitivity curve</td>
<td>61</td>
</tr>
<tr>
<td>3.15</td>
<td>Long term drift response and the effects of resetting</td>
<td>61</td>
</tr>
<tr>
<td>3.16</td>
<td>Measured output noise power spectrum</td>
<td>63</td>
</tr>
<tr>
<td>3.17</td>
<td>Transient response to pH variation</td>
<td>63</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

3.18 Chemical reaction monitoring of a pH diffusion process .......................... 64
3.19 ISFET transistor under the effect of switch when it is multiplexed off ........ 66
3.20 Schematic of proposed single stage front-end ........................................... 67
3.21 Microphotograph of the fabricated array .................................................... 67
3.22 Gain Distribution and its cumulative counts ............................................... 68
3.23 Transient response of pH step change ....................................................... 69
3.24 Measured noise of single stage front-end ................................................... 70
3.25 Schematic of proposed 3-T front-end and its simplified model .................... 71
3.26 Microphotograph of the fabricated array ................................................... 72
3.27 Test of Amplification .................................................................................. 73
3.28 Step output changes from pH 7 .................................................................. 74
3.29 Sensor output noise spectrum ................................................................. 74

4.1 Concept of gain calibration ......................................................................... 80
4.2 Diagram of the automatic gain control system ............................................ 82
4.3 Flow diagram of the signal path in the AGC .............................................. 82
4.4 Diagram of system without log amplifier ................................................... 84
4.5 Diagram of the amplitude detector ............................................................. 85
4.6 Limitation of time constant due to SNR requirement .................................. 86
4.7 Transient step response of a single pole system ........................................ 87
4.8 Diagram of proposed system with additional lowpass filter ....................... 87
4.9 The influence of adding another pole .......................................................... 89
4.10 Transient step response of a tri-pole system ............................................. 91
4.11 Effect of the offset in rectification ............................................................. 92
4.12 Noise path in the calibration system ................................. 93
4.13 Diagram of implementation of AGC system .......................... 94
4.14 Block level structure of the variable gain amplifier ................ 95
4.15 Schematic of the transconductor used for VGA ...................... 97
4.16 Schematic of Pre-amplifier with two highpass filter ................ 98
4.17 Block structure of the amplitude detector ............................ 98
4.18 Schematic of resistive voltage-to-current converter .................. 99
4.19 Schematic of phase generator ...................................... 100
4.20 Basic current mode lowpass filter by dual-output OTA-C structure .... 101
4.21 The layout of proposed system including output buffers .......... 102
4.22 Tested control voltage under constant input ........................ 103
4.23 The spectrum of tested control voltage .............................. 103
4.24 Tested transient results of the control voltage and output voltage .... 104
4.25 Differential control voltage for a range of input amplitude ......... 104

5.1 System Architecture .................................................... 110
5.2 Pixel array readout architecture .................................... 111
5.3 Implementation of the automatic gain calibration system ........... 112
5.4 Timing graph of digital control logic ................................ 113
5.5 Schematic of pixel in the array ...................................... 114
5.6 Layout of pixel in the array ........................................... 115
5.7 6th-order lowpass filter consists six 1st-order gm-C filters .......... 115
5.8 Structure of an Instrumentation Amplifier ............................ 116
5.9 The layout, photo and floor plan of the fabricated chip ............ 117
LIST OF FIGURES

5.10 Test set-up for the system ........................................... 118
5.11 The timing chart of the operation of the whole system ............ 120
5.12 The final output values for different pixels .......................... 121
5.13 The distribution of the DC voltages over 4 chips ................... 121
5.14 Output drifts due to the switch of the whole array .................. 122
5.15 Distribution of drift rate due to switch leakage of the whole chip . 122
5.16 pH image of the whole sensor array from pH 9 to pH 7 .......... 123
5.17 Voltage outputs of the array before and after the flowing of the solution ........................................ 124
5.18 Sensitivity deviation due to corrosion using Sodium Hydroxide 125
5.19 Amplification test shows a good consistence over the whole chip 126
5.20 The distribution of the gain over 4 chips ............................. 126

6.1 Assembly produced by the greedy approach ........................ 131
6.2 A String Graph representation and its reduction ................... 133
6.3 Overlap graph of a genome containing a two-copy repeats ......... 133
6.4 Schema of assembly in de Bruijn method ............................. 134
6.5 Mechanism of dynamic suffix comparison ............................ 138
6.6 Table for dynamic programming calculation .......................... 140
6.7 Partition Dynamic Programming and its shift mechanism .......... 141
6.8 shift compensation mechanism ....................................... 143
6.9 Simulation results for different block length and redundancy .... 144
6.10 Array and pixel architecture ........................................ 146
6.11 Architecture of the proposed system .................................. 147
6.12 Timing chart of system indicating the processing operations .... 148
6.13 Infix retrieval from different locations ........................................ 150
6.14 The composition of an overlap hitting in the FIFO buffer ................ 151
6.15 Systolic array with AUs used for implementing dynamic programming ... 152
6.16 Hardware Implementation of Dynamic Programming .......................... 152
6.17 Timing diagram showing the parallel assembly process with detection ... 155

B.1 Dolphin – 32x32 ISFET array system .............................................. 167
B.2 Panda – 32x32 ISFET array system ................................................. 168
B.3 Altair – ISFET characterisation devices and Automatic Gain Control System 168
B.4 Gonzo – ISFET characterisation devices ........................................... 168
B.5 Kimi – ISFET characterisation devices ............................................. 169
B.6 Kermit – ISFET characterisation devices ........................................... 169

C.1 Demonstration of two DNA detection methods relying on physical changes . 170
C.2 The approximation model of DNA strands attached on the electrode ........ 171

D.1 The hierarchy of entity instantiation ...................................................... 174

F.1 Schematic of test circuits PCB ......................................................... 211
F.2 Layout of test circuits PCB .............................................................. 212
List of Tables

1.1 Consequence of the drawbacks of ISFET ................................. 4

2.1 Summary of mentioned sequencing technologies .......................... 21
2.2 Comparison of CMOS ISFET arrays over the last decade ............... 29
2.3 Required specification of ISFET front-end for efficient DNA detection .... 35

3.1 Design trade-off for different applications ............................... 54
3.2 List of fabricated ISFET devices under test .............................. 55
3.3 Individual Die statistics for $C_f=10\text{fF}$ ............................. 58
3.4 Summary of amplification measurements over all dies ..................... 59
3.5 Voltage variation corresponding to pH changes ........................... 60
3.6 Achieved system specification ............................................ 65
3.7 Comparison of ISFET readouts with integrated compensation schemes .... 65
3.8 Specifications of the single-stage interface .............................. 70
3.9 Summary of measured amplification over all chips ........................ 73
3.10 Specifications of the proposed interface ................................ 75
3.11 Comparison of three proposed front-ends ................................ 76

4.1 Specification of the transconductor ...................................... 96
### LIST OF TABLES

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Specification of the transconductor</td>
</tr>
<tr>
<td>4.3</td>
<td>Designed specification of the whole AGC system</td>
</tr>
<tr>
<td>5.1</td>
<td>Standard deviation of offset voltages over chips</td>
</tr>
<tr>
<td>5.2</td>
<td>Standard deviation of the gain of pixels over chips</td>
</tr>
<tr>
<td>5.3</td>
<td>Specifications of the full system</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of approaches for implementation in FPGA</td>
</tr>
<tr>
<td>6.2</td>
<td>Parameter settings for FPGA Implementation</td>
</tr>
<tr>
<td>6.3</td>
<td>FPGA Resource Usage</td>
</tr>
<tr>
<td>6.4</td>
<td>Overlap hitting found by proposed algorithm for three bacterial genomes</td>
</tr>
</tbody>
</table>
Dedicated to my parents!
Acknowledgement

First and foremost I would like to express my deepest thanks to my supervisor Dr. Pantelis Georgiou for his motivation, support and guidance throughout my PhD. Working with Pantelis has been an outstanding endeavour, as his inspiring thought largely widen my horizon and his firm support raised me up over the depressed days. It has been an honour to be his student and a member of his team here at the Centre for Bio-Inspired Technology.

I also would thank Dr Timothy Constandinou for his countless discussions and advice, thank Pantelis Athanasiou from DNAe for his help in designing and producing the flow cell for chemical tests, and thank Dora Ma for her generous proof reading. Special mention must go to Ms Maria Khaleeq from the Department of Physics for her warmth, supportive aura and willingness to help in any chip-related post processing step.

Moreover, I would like to express my gratitude to my colleagues and friends in the CBIT for their outstanding love and support: Konstantin, Andrea, Amir, Pau, Melina, Yan, Song, Nicoletta, Sivylla, Irina, Ian, Claudio, Deren, Mohamed, Ermis, Lieuwe, Dora, Nick, Peter, Mo, Mohammadreza, Francesca ... and the list goes on.

Having been the UK for 5 years but I never felt lonely in another country. I have to credit this to my all Chinese friends. Although we are scattered around the world, I would like to express my deeply appreciation to them and firmly believe the friendship will last forever.

Most of all, I thank my parents for all their love, unconditional support and continual encouragement. This journey would not have reached its destination without them. It is their education and influence that shape my character today. Nowadays I cannot be more recognised to myself ever in my life and this four years experience of both studying and living finally eliminates my inner hesitation. I could expansively envisage my desire and decently accept any results. I dedicate this thesis to them.
Chapter 1

Introduction

1.1 Motivation

Ever since it was discovered that DNA\(^1\) contained heredity information [1], DNA sequence data has been decoded and analysed extensively. Due to the significance of the information carried by DNA, its study has led to various research areas such as Genetics, Bioinformatics and DNA Nanotechnology. The understanding of the properties of DNA sequences has been beneficial for a variety of applications including but not limited to cancer analysis [2], infectious diseases [3], human genetics [4], personal genomes studies [5] and even ecology [6]. For example, investigating genetic defects through DNA allows for more precise identification of the causes of certain cancers which then facilitates the early detection before metastasis and treatment with the right drugs [7]. As a result, there is a desire to have faster, more reliable and affordable DNA sequencing systems which can give us genetic data to further analyse and improve health care.

The growing need to obtain large amounts of DNA data for various applications has motivated efforts to drop the cost and time of sequencing. The completion of the Human Genome Project in 2003 based on the traditional Sanger sequencing [8, 9] was a great achievement of an international effort over 13 years, which costs an estimated 300 million US Dollars. In contrast, with the achievement of technology, a company called Life Technologies is now claiming they are making a DNA sequencer expected to sequence the entire human genome in a day for just $1,000 [10]. This tremendous decrease has been tracked by the National Human Genome Research Institute (NHGRI), whose findings show that the trend

\(^1\)DNA stands for DeoxyriboNucleic Acid, discovered in 1952
in cost reduction in DNA sequencing follows more than an exponential drop [11].

The main reason for this dramatic price drop is attributed to a new massively parallel sequencing method, which is termed Next-Generation Sequencing (NGS). The main characteristic of NGS is its capability to produce millions of DNA sequences concurrently [12]. Since the emergence of the first commercial application of this technology in 2004, there have been many technologies developed applying this technology [13]. Among them are technologies with great potential, which are now commercial: Ion semiconductor (Ion Torrent), Pyrosequencing (454), Sequencing by synthesis (Illumina) and Sequencing by ligation (SOLiD sequencing) [14, 15].

The majority of existing sequencing methods are based on optical detection, which require imaging technology and some special reagents [16]. On the other hand, the emerging semiconductor based sequencing techniques [17], enabled by the use of ISFETs (Ion Sensitive Field Effect Transistors), benefit from semiconductor industry’s Moore’s Law [18] to increase the number of sensing sites as the technology scales. Additionally it allows for the integration of sensor instrumentation to achieve high range SNR, high speed and low cost data readout. All these features makes semiconductor based sequencing an attractive research domain.

ISFET based sequencing systems also suffer from problems such as offsets caused by trapped charge, sensitivity loss due to capacitive division, and drift as a result of the chemical reaction on the sensor surface. The direct consequence of these problems to the overall sensing system makes their operation suboptimal.

<table>
<thead>
<tr>
<th>Drawbacks</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Readout offset</td>
<td>higher resolution of ADC required</td>
</tr>
<tr>
<td>Sensitivity loss</td>
<td>Auxiliary circuit overhead required</td>
</tr>
<tr>
<td>Drift</td>
<td>higher resolution of ADC required</td>
</tr>
</tbody>
</table>

As shown in table 1.1, due to the offset and drift of the sensed output, a larger resolution of analogue-to-digital converter is required while the sensitivity loss requires auxiliary circuits to bootstrap the parasitic capacitors. As a result, current systems need fast and accurate ADCs in addition to fast peripheral readout circuits to send out this large sampled data with a sufficient frame rate. Improvements on the sensor itself could largely alleviate the design complexity of other subsystems. There is potential to use conventional analogue techniques
in an array to remove all these problems and capture just the reaction as it occurs in a reliable way. In doing so it will reduce the required ADC resolution for detection and relax the requirement needed for the system.

As mentioned, one advantage of ISFET sensors is their ability to scale with Moore’s Law, increasing pixel density in arrays as technology scales. Following this trend however can be particularly challenging when implementing ISFETs in unmodified CMOS. Along with the process improvement, transistor size will decrease progressively, however the passivation thickness of ISFET sensors does not change proportionally. As a result, if we scale the pixel size with the process variation, the absolute passivation capacitance would be reduced, worsening the sensitivity loss issue due to capacitive division. To guarantee scaling instrumentation needs to be designed which prevents this sensitivity loss.

Another inevitable problem when conducting DNA sequencing is that the detected DNA strands are limited in length to approximately up to 700 base pairs (200 bp only in Ion semiconductor), which is orders shorter than the original DNA. For example, the largest human chromosome, chromosome #1, is about 220 million base pairs long [19]. To restore the required original DNA data, a number of assembly algorithms have been developed, categorized into three types: Greedy, Overlap-Layout-Consensus and De Bruijn [20, 21]. Assembly algorithms take advantage of the overlaps between short sequences which is analogous to reconstruct an original signal from oversampled data set. But assembly in itself is also a computational intensive problem.

As a result, the typical NGS sequencing of DNA requires detection, readout and assembly in three distinctive steps, whereby detection could be based on optical or semi-conductor
sensors and assembly conducted on general work stations, shown in figure 1.1.

Furthermore, NGS systems may take a long time to execute a sequencing process, from 2 hours (Ion semiconductor) to 2 weeks (Sequencing by ligation). Specifically semiconductor sequencing, a sequencing-by-synthesis method, takes advantage of the fact that DNA polymerizations happen only in complementary pairs (A-T or C-G). Whenever a polymerization takes place, there is a release of hydrogen ions whose concentration can be monitored by pH sensors (ISFET). The system uses a test flow whereby every Nucleotide (A,T,C,G) is introduced sequentially, to incorporate the DNA strand under test. Each insertion of Nucleotide flow however takes at least 2 seconds, which is a huge time span that potentially can be utilised by very fast microelectronic systems. Hence there is an opportunity to utilise this interval time to do parallel comparison in hardware using an FPGA (Field-Programmable Gate Array) platform which would ultimately speed up the assembly process.

1.2 Research Objectives

This research aims to investigate a method for a full hardware based solution for parallel detection using ISFET sensors and assembly of DNA fragments in FPGA. Particularly we aim to design circuits which can address the drawbacks in current ISFET sensors shown in table 1.1, such as drift, trapped charge and sensitivity loss. The proposed ISFET sensor implementation should be scalable to a large array which enables the design of NGS systems. An ISFET sensor array may also suffer from the gain mismatch in every single pixel, thus an auxiliary system to calibrate them is desirable. An automatic gain control system specifically designed for this is implemented as a solution.

In terms of DNA assembly work, research investigating the feasibility of the FPGA implementation of assembly algorithms is undertaken. Certain modification of existing algorithms is expected to utilize the sequencing time. Finally the parallelization of this computational-intensive assembly process is realized in FPGA.

The vision of this work is to ultimately contribute to a novel system shown in figure 1.2 to be developed. The detection, readout and assembly are expected to execute concurrently. They are also all implemented in semiconductor technology.

Subsequently, this work is targeted towards providing some insight into the following:

- Can a new sensing methodology for ISFET reaction monitoring immune to non-ideal
"Introduction"

Figure 1.2: Flow for the proposed hardware based DNA sequencing system

effects such as trapped charge, capacitive division and drift be developed?

- Can the new ISFET front-end be used in the implementation of large sensing arrays??
- Can a novel AGC system for chemical sensors be applied to eliminate the pixel sensitivity mismatch?
- Can a practical real-time DNA comparison algorithm be developed and implemented in an FPGA which can be run during the detection and be used for faster fragment assembly?

1.3 Thesis Organisation

This thesis is organised into five main chapters presented as below:

1.3.1 DNA sequencing using semiconductors

The second chapter first introduces the general background about popular DNA sequencing technologies and the new semiconductor approach. The theory behind the capability of chemical sensing using semiconductors is then explained. Existing implementation of sensors such as ISFET is then detailed followed by its current challenges.
1.3.2 ISFET interface based on capacitive feedback

This chapter introduces a new ISFET sensor topology capable of overcoming all the existing drawbacks, such as drift, trapped charge and capacitive division. The key in our interface is to use capacitive feedback and two different working phases for the ISFET: a resetting phase and sensing phase. As their names imply, in the resetting phase the output is cleared of any error, while in the sensing phase it responds to the input reaction. Design considerations are introduced which may facilitate others to apply this structure in other applications. Following the same concept, a single-stage front-end has also been investigated where a single transistor is used as the amplifier. Finally, an extreme case where parasitic capacitor serves as the feedback capacitor is studied, and the features of these three topologies have been compared.

1.3.3 Automatic Gain Control Systems for ISFET array compensation

This chapter focuses on an automatic gain control system which is specifically designed for our ISFET front-end array. A high-frequency small-amplitude sinusoidal wave is superimposed on the chemical signal to be used as a reference to all the pixels. The AGC system utilizes this sinusoidal wave to adjust the gain of each ISFET sensor. Considering its unique application, several specifications have to be emphasised, such as fast tracking and high accuracy. As a result, a linear-in-dB AGC is firstly applied. Then its tracking speed and accuracy trade-off is found and analysed, and the theoretical limitation is pointed out. An approach which inserts additional filters allows the AGC to function beyond the initial limitation, which is also proven by tested results. A realization of each sub-blocks comprising the total AGC system is presented.

1.3.4 An ISFET Chemical Sensing Array with Automatic Gain Calibration

In this chapter, we are going to demonstrate the implementation of a large array with the ISFET sensor and instrumentation. Because of the long period of chemical reaction monitoring, digital CDS (correlated double sampling) technique is applied. To store these large amounts of digitalized data, a real time SPI (Serial Peripheral Interface) is implemented to store the offset data off-chip. A $32 \times 32$ array along with the necessary logic circuits has been designed and fabricated in CMOS. The effect of the calibration system is discussed.
1.3.5 Real-Time Sequence Comparison for DNA Assembly

This chapter includes work conducted for sequence assembly in hardware. We first review and compare the existing assembly algorithms and find the Overlap-Layout-Consensus (OLC) method to be the most suitable one for our parallel implementation. In terms of the OLC approach, the first step is finding the overlaps which is the most time-consuming part, making it the most worthy part to be implemented in FPGA. As we wish to utilize the sequencing time to execute our assembly processing, a hybrid overlap searching algorithm is applied which is scalable and can deal with incremental detection of new bases. To handle the incomplete data set which gradually increases during the sequencing time, all-against-all comparisons are broken down into successive window-against-window comparison phases and executed using a novel dynamic suffix comparison algorithm in combination with a partitioned dynamic programming method. The complete system has been designed to facilitate parallel processing in hardware, which allows real-time comparison and complete scalability. Implemented on an FPGA, a base pair comparison rate of $51.2G/s$ is achieved and the DNA assembly of fragments from real genomes is proved.
Bibliography


Chapter 2

DNA Sequencing methods and the semiconductor based approach

2.1 Introduction

DNA which stands for DeoxyriboNucleic Acid contains the genetic instructions that steer the development and functioning of almost all living organisms. Since its role in carrying heredity information was confirmed in 1952 [1], DNA sequences have been decoded and analysed extensively, and the operation of its biological functions has led to a huge demand for understanding and research.

Though the majority of DNA sequence production is generated by Sanger biochemistry or its variants [2], the urgent and intense requirement of massive and economical DNA sequencing technology has driven people’s interests towards Next-Generation Sequencing (NGS) [3–9]. NGS has resulted in a tremendous increase in the amount of sequenced data and a drop in cost exponentially [10].

Evidently the key advantage of NGS technologies is its capability for massive parallel detection. For instance, Ion Torrent chip semiconductor technology [11] can now integrate millions of sensors on a single microchip to detect 100 million DNA fragments in parallel. In general the cost is lower credit to three advantages of the NGS platform: 1) in vitro library construction and clonal amplification, 2) parallel sequencing via large sensing array and 3) effective reagent manipulation to the scale of picoliters or even femtoliters per feature.

Among these new technologies, semiconductor-based detection method using ISFET
sensors removes the necessity of fluorescence and imaging detection, and achieves even higher speed and lower cost. The latest sequencing chip from Ion Torrent could generate 60-80 millions reads in only 2-4 hours [12]. More importantly semiconductor approaches do not only benefit from the scaling trend under Moore’s Law [13], but also compatibility with other peripheral readout and signal processing circuits such as gain calibration circuits introduced in this thesis.

In this chapter, the background of DNA sequencing and its semiconductor approach will be introduced to give an understanding of how these areas of DNA sequencing and semiconductor can be bridged.

2.2 DNA properties and base pairing

DNA is composed of repeating nucleotides forming a long polymer [14]. The well-known DNA double-helix model was first introduced in 1953 by James D. Watson and Francis Crick [15], shown in figure 2.1(a). Every single DNA strand contains two parts: the Backbone that holds the chain together, and Nucleobases which are responsible for the bonding with other DNA strands in the helix.

![Figure 2.1: Double-Helix model of DNA and its chemical structure](image)
There are four types of bases found in DNA, they are Adenine (Abbreviated A), Cytosine (C), Guanine (G) and Thymine (T). In the double-helix structure, each base from one strand can only incorporate with a specific base on the other strand, e.g. A-T and C-G, which is called complementary base pairing shown in figure 2.1(b). Base pairs are held together by the hydrogen bond which is not as strong as covalent, hence it is feasible for two strands to be broken and rejoined. The two properties mentioned above are essential in diversifying DNA detection methods.

In the backbone of DNA strands, Deoxyribose (a 5-carbon sugar in blue colour) are linked together by phosphate groups forming phosphodiester bonds between the third and fifth carbon atoms of two adjacent carbon rings respectively, shown in figure 2.1(c). Due to this asymmetric connection, DNA strands are designated to be directed, and in one double helix the direction of the two strands are opposite to each other. We term the terminal having the phosphate group to be the 5’ end (five prime) and the one with the hydroxyl group as the 3’ end (three prime).

2.3 DNA incorporation, hybridisation and sequencing

![Figure 2.2: Biochemical mechanism of Nucleotide incorporation which release hydrogen ions and phosphate groups](image)

The DNA strand is replicated by the chain extension of a complimentary base pair incorporation. When the correct dNTPs (deoxynucleotide triphosphates, containing four types dATP, dTTP, dCTP and dGTP) are added for incorporation, there is nucleophilic
attack on the α-phosphate group from the 3’end hydroxyl group of the last nucleotide of the growing DNA strand [16]. As shown in figure 2.2, the process results in a production of two phosphate groups (β-& γ-phosphate) as well as a hydrogen ions (H\(^{+}\)), causing a pH change in solution. This gives opportunities for developing label-free DNA detection methods.

There are several different approaches to detect DNA fragments, which all utilise the fact that Nucleotide incorporation can take place only between complimentary bases. Sequencing-by-hybridisation monitors binding of one strand of DNA to its complementary strand, where one of them is previously known as prime. This kind of binding will cause a change in terms of mass, viscosity or stress and therefore can be detected. In addition to that, the binding can also be detected through electrochemical approaches or fluorescent tags which requires the DNA to be labelled such that the binding can be detected. Methods such as cantilever-based and resonance-based approaches which detect DNA hybridisation are explained in Appendix C.

For DNA sequencing, however, detection has been largely executed through other methods such as sequencing-by-synthesis and sequencing-by-ligation which will be described in the following subsections. In the following section, we show that both the phosphate groups and hydrogen ions can be utilised to detect the incorporation process. It is also known that DNA polymerase are also acceptable to modified nucleobases which extends a large group from their base body [17, 18]. Hence, dNTPs and their modifications such as dNTP(AP\(_{3}\))-SS-AF594 [19] can also be used to trigger the incorporation.

### 2.4 Next generation DNA sequencing

Before we introduce the various sequencing technologies, it is worth noting that all methods described would require DNA clonal amplification to magnify the signal strength due to the minimal amount of phosphate group or hydrogen ions released from a single incorporation. The amplification, more specifically Polymerase Chain Reaction (PCR), can be realised by either emulsion PCR [20] or bridge PCR [21, 22] depending on the sequencing technique applied. Other approaches that do not rely on the PCR process such as Nanopore are beyond the scope of this thesis, interested readers can refer to [23-25].
2.4.1 454 Pyrosequencing

The 454 system is the first commercial NGS product in 2005 [26]. This technique is based on the Pyrosequencing method which detects the pyrophosphate released during nucleotide incorporation [27]. As shown in figure 2.3, the beads binding with multiple identical DNA signal strands are immobilised in a micro-fabricated well array.

![Figure 2.3: DNA bonded beads immobilised in the well](image)

The whole detection process comprises several hundred cycles; at each cycle one type of dNTP (dATPs, dCTPs, dGTPs and dTTPs) combines with the necessary enzyme and luciferin are introduced to all the wells in the sensing array. In the wells where incorporation happens, pyrophosphate is generated and converted into ATP (Adenosine triphosphate) catalysed by ATP sulfurylase. This ATP could further convert luciferin into oxyluciferin where visible light is generated, whose strength is proportional to the amount of ATP present. Hence an optical-based detection can be realised. The un-incorporated dNTPs and ATP are degraded by Apyrase after every cycle, which is called an Apyrase wash.

As shown in figure 2.4, each flow cycle only introduces one of four different dNTPs. A primer consisting of a series of complementary bases to the start of the target DNA is used to initiate DNA strand extension. Because the Primer is known, its chemical response amount from Nucleotide incorporation can also be set as a reference to normalise the following reaction. This allows for quantifying the amount of polymerisation, which gives an indication of how many bases are incorporated in each reaction. For example, the first two ‘A’ in the strand do not react to the insertion of ‘A’ (dATPs) in the flow but do react to the ‘T’ (dTTPs) with a signal twice as large as the threshold. We can then derive that this sequence starts with two ‘A’s, followed by a C since the next response takes place during the introduction
of ‘G’ (dGTPs). The absence of terminating moiety which prevents multiple consecutive incorporations makes it works asynchronously as homopolymers (consecutive instance of the same bases, e.g. AAA) will be detected in one cycle. Take the case above as an example, two ‘A’s have been detected at the same time.

### 2.4.2 Illumina/Solexa system

The Solexa system was first released in 2006 and was purchased by Illumina one year later in 2007 [22]. Instead of introducing dNTPs, this system uses synthesised analogs of nucleotides (dNTP(AP₃)-SS-AF594), where fluorophore is linked to the 5-position of pyrimidines (C and T) and 7-position of purines (A and G). In addition to the fluorescence, this fluorophore could also terminate the strand elongation due to its steric hindrance [28], which makes these nucleotide analogs act as terminators. The fluorescence is also designed to be cleavable so that the incorporation can be resumed one after the other [19].

As figure 2.5 shows, four types of modified deoxynucleotide species are added at the same time and only one nucleotide extension is allowed. The sequencing is implemented by distinguishing different colours from four different species. Only when the fluorophore is chemically cleaved away can another incorporation be executed. Ideally, Solexa system performs a synchronous sequencing as all the strands extend in the same phase, whereas in reality phase offset always takes place.
2.4.3 AB SOLiD

The SOLiD, which stands for Sequencing by Oligo Ligation Detection, was first introduced in 2005 [29] and purchased by Applied Biosystems in 2006. Unlike other sequencing technologies, it adopts the ligase rather than the polymerase\footnote{Different enzymes are needed for ligation and polymerisation}. Each ligation probe contains 8 bases including two ligation sites, cleavage site and fluorescent label. The sequencing process is shown in figure 2.6.

Each detection cycle introduces a degenerated population (4 out of 16) of fluorescently labelled probes (a). Only the probe with two bases complementary to the template stand could trigger the ligation, which can be optically detected (b). Then the last three bases (zzz in figure) are chemically cleaved (c), which eliminates the previous fluorescent signal and leaves a free end for next ligation (d). Iterative cycles of these ligation, detection and cleavage could detect an evenly spaced set of bases (e.g. bases 1,2,6,7,11,12). After completing several cycles, the template is reset and the process is restarted again for another round with one-offset primer. Figure 2.7 shows the detection diagram after 5 rounds where every bases has been sensed twice independently.

Clearly due to the lack of probes (4 out of 16), the method is only suitable for resequencing rather than \textit{de novo} sequencing.
2.4.4 Ion Personal Genome Machine

The Ion PGM platform was released by Ion Torrent at the end of 2010 and is the first commercial sequencing technology that does not require any labelling or luciferase which is the fundamental element in other optic-based technology.

It shares similar detection methodology as Pyrosequencing mentioned before. Nucleotides
are introduced to the chip one after another, and only complementary bases will trigger an incorporation event whose response strength relates to how many Nucleotides are involved. The main difference from Pyrosequencing is that Ion PGM detects the Hydrogen ions rather than pyrophosphate released in the incorporation process. There is approximate 0.02 pH change per single base incorporation [30]. As long as there are enough copies of the target DNA strands, the pH change will be large enough to be detected electronically.

Figure 2.8 shows the signal pattern of 1-nucleotide incorporation, where the blue line is the background corrected data and red line the physical model. The incorporation would trigger a signal jump depending on the reaction level. Then the signal decays because of the chemical washes after every flow insertion.
2.4.5 Summary of different sequencing technologies

A detailed comparison has been presented many times in several reports [3, 5, 6, 8, 9]. Usually the technologies can be configured for different applications depending on the required specifications. Take the Illumina platform as an example, the rapid mode could sequence 25 million reads with lengths of 300 base-pair in 65 hours, whereas high output mode detects 3 billion reads at each run, but with lengths of only 100 base-pair and takes 12 days. Hence it is not appropriate to compare them directly, but to summarise their representative features here in terms of speed, accuracy, cost etc. Some specifications are listed in table 2.1.

Table 2.1: Summary of mentioned sequencing technologies

<table>
<thead>
<tr>
<th>Method</th>
<th>454-Pyrosequencing</th>
<th>Illumina/Solexa</th>
<th>AB SOLiD</th>
<th>Ion Torrent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Length</td>
<td>700 bp</td>
<td>up to 300 bp</td>
<td>50±50 bp</td>
<td>up to 400 bp</td>
</tr>
<tr>
<td>Accuracy</td>
<td>99.9%</td>
<td>98%</td>
<td>99.99%</td>
<td>98%</td>
</tr>
<tr>
<td>Time per run</td>
<td>23 h</td>
<td>65 h to 12 d</td>
<td>10 d</td>
<td>4 h</td>
</tr>
<tr>
<td>Output per run</td>
<td>700 Mb</td>
<td>up to 600 Gb</td>
<td>300 Gb</td>
<td>10 Gb</td>
</tr>
<tr>
<td>Cost/10^6-bases</td>
<td>$10</td>
<td>$0.07</td>
<td>$0.13</td>
<td>$1</td>
</tr>
<tr>
<td>Advantages</td>
<td>read-length, fast</td>
<td>low cost</td>
<td>low cost, accurate</td>
<td>less expensive instrument, fast</td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Homopolymer errors, costly</td>
<td>expensive instrument, errors</td>
<td>expensive instrument, resequencing only</td>
<td>Homopolymer errors</td>
</tr>
</tbody>
</table>

For the optical-based system, the most time consuming process is the imaging of flow-cell. Sequencing time could be reduced by sacrificing the imaging area and its corresponding data, which is the case in Illumina system. Ion Torrent technology, however, completely eliminates the necessity of imaging detection and as a result is able to finish each single run in 4 hours, making it the fastest method. Pyrosequencing is relatively faster than other two due to its succinct detection steps.

Among the above mentioned technologies, AB SOLiD detects each base twice independently thanks to its two-base interrogation scheme and hence achieves very high accuracy (99.94% for raw data). However, the degenerated population of probes hinders it from the de novo sequencing applications. Both Pyrosequencing and Ion PGM suffer from the insertion/deletion error in homopolymers, especially for homopolymers longer than 6 bp, due to the absence of terminator. Although the Solexa system can prevent these errors by allowing...
DNA Sequencing methods and the semiconductor based approach

only one incorporation each time, other factors such as incomplete cleavage of fluorescent label still hamper the sequencing accuracy. Nowadays all the platforms apply different error filtering or correction techniques and the data shown in table 2.1 is the corresponding achieved figures.

In terms of the cost, it is usually inversely proportional to the productivity. Currently the number of reads that can be detected concurrently per run is limited by the dimension of beads used for PCR process [31]. For example, the beads used in 454 Pyrosequencing is 28μm whilst the beads applied in SOLiD is only 1μm, which allows it to output more data as a result of high density. Illumina system bypasses this problem by adopting bridge PCR to achieve even higher production, and is currently the cheapest sequencing technology.

Thanks to the large scale integration of semiconductor, the recently announced Proton-III chip from Ion Torrent increases the well capacity to 1.2 billion and will be able to generate as much data as Illumina. In addition, avoiding the use of bulky optical instruments not only drops the total cost, but also allows for miniaturisation of devices. Finally the compatibility with other peripheral readout and signal processing circuits makes this semiconductor approach more promising in realizing higher speed and lower cost.

In the next section, chemical sensing in semiconductor will be introduced.

2.5 Semiconductor based chemical sensors

Chemical sensing using semiconductors relies on the charge-transfer reactions and is usually realised in three different approaches: voltammetric, potentiometric and conductimetric [32]. All these methods requires certain post-processing after fabrication, either by depositing electrodes on the die or attaching ion-selective membrane according to the chemicals to be sensed. The principles of three approaches are briefly described below:

Voltammetric (or Amperimetric) sensors measure the basic current-voltage relationship in targeted chemical environment. Given a constant potential between two electrodes, the current flow has a linear relationship with the analyte concentration according to Cottrel equation [33, 34]:

\[ I = n_e \cdot F \cdot A \cdot D_{\text{diff}} \cdot \frac{c_A}{L_{\text{diff}}} \]  

(2.1)

where \( n_e \) is the number of electrons, \( F \) the Faraday constant, \( A \) the effective electrode area, \( D_{\text{diff}} \) the diffusion coefficient, \( c_A \) the target analyte concentration and \( L_{\text{diff}} \) the diffusion
length. As can be seen that the measured current depends not only on the analyte itself but also the composition and geometry of electrodes. Usually an additional auxiliary electrode accompanied with an OpAmp is required to prevent the reference electrode from carrying current. Planer electrodes on CMOS substrate requires post-processing of inert metals deposits such as platinum or gold.

Conductimetric sensors, including chemiresistor and chemicapacitor, measure the impedance of the sample determined by the concentration of the analyte. If only the resistive part of the impedance is of interest, resistance measurement arrangement such as Wheatstone bridge can be set up. This type of sensor is suitable for large array systems but usually cannot provide decent sensitivity because of the negative impact when the sensor are downsized.

The potentiometric sensor measures the potential between two electrodes immersed in analyte based on Nernst equation [35]. The two electrodes comprise a reference silver/silver chloride (Ag/AgCl) electrode and a working electrode covered in ion-selective membranes. The field-effect-based potentiometric devices, such as MOSFETs, fall into this class. They can sense the charge distribution on the semiconductor surface and consequently measure the concentration of the analyte. When the equilibrium has been reached, there is no current flow in the analyte.

In general, voltammetric and conductimetric approaches benefit in miniaturization due to their simple realization. However they suffer from noise and poor SNR performance because of the inherent thermal noise from resistors [36]. Another problem of Voltammetric and conductimetric approaches is their broad selectiveness, in other words, they are sensitive to excessive chemicals.

Potentiometric sensors have a particular advantage in terms of miniaturization since the integrity of the chemical signal does not depend on the sensing area [36]. Despite the necessity of an external reference electrode, which limits the miniaturization of ENFETs (enzyme FET), ISFETs (ion selective FET) have experienced significant progress due to the improvement of the ion selective reference electrode.

Among various implementations, the Ion Sensitive Field Effect Transistor (ISFET), belonging to the potentiometric class, can be fabricated in standard MOSFET process and only needs one electrode to supply external reference. This comes from the merit that the native insulating layers of CMOS process, for example Si$_3$N$_4$, is sensitive to the Hydrogen ions [37]. Because of this possibility of monolithic integration of both ISFET and MOSFET,
using ISFETs as chemical sensors have more potential for further miniaturisation and lower cost.

2.6 Ion Sensitive Field Effect Transistor

The initial motivation for creating ISFETs came from the need to measure sodium and/or potassium ions in certain area of the thalamus and at the same time be immune to the influence of all other potential changes caused by factors such as electrochemical processes [38]. Compared to the conventional glass membrane electrodes, ISFET based sensors can be easily miniaturised, whereas products applying the former technology are prone to be unstable and fragile when their sizes are shrunk. Additionally, ISFET technology, belonging to the solid state devices, could respond to input changes much faster than glass electrodes. All these merits make ISFET sensor appealing and successful in the biomedical domain.

The selectivity and sensitivity of ISFETs are determined by the interface between the chemicals and semiconductors, which serves as the gate insulator too. Due to the limitations in deposition technologies, the initial research and design of ISFETs used intrinsic materials such as $Si_3N_4$ and $Al_2O_3$. Afterwards many combinations were experimented empirically to increase the sensitivity without the support of theory until the discovery of the significance of $[H^+]_s$ in 1995 [39]. In addition to the investigation of the gate insulator, sensor encapsulation is also a topic of interest to protect the metal pads from the solution whilst allowing the semiconductor sensing area to be exposed. [40-42] are works dedicated to address this problem.

In the new era of ISFETs, the sensors are implemented in large arrays for massive parallel sensing. In [43] a summary of the current outstanding works which managed to integrate multiple sensors, readout circuits, signal processing units and other auxiliary circuits into a single chip was exhibited. This success makes ISFETs particularly appealing in certain applications such as DNA sensing where large amount of data outputs are desired. In the following section, the sensing mechanism of the ISFET will be discussed.

2.7 ISFET sensing mechanism

The ISFET is a modification of the original MOSFET by replacing its gate metal with electrolyte solution in contact with a reference electrode. The gate insulator, e.g. silicon
dioxide, is directly exposed in the solution and the reference electrode is used to supply a potential to the gate, shown in figure 2.9. Due to its potential for large-scale integration in addition to fabrication using an unmodified CMOS process, it has recently been adopted in various lab-on-chip and health-care applications [44] [37].

![Figure 2.9: The MOSFET and the ISFET](image)

The capability of pH sensing for the insulating layer can be described through the Electrolyte-Insulator-Semiconductor boundary [45] which consists of two phenomena: the site binding of ions at the insulator surface [46] and the Helmholtz & Gouy-Chapman capacitive double layers in solution [47]. More details are given in the following subsections.

### 2.7.1 Electrolyte-Insulator Interface

According to the site binding model [46], the surface of the insulator will react with the ions in the solution, leading to the existence of ion donors or ion acceptors whose charges could exhibit a potential change upon the burying semiconductor.

Take the silicon dioxide as an example in figure 2.10, there are three types of sites: \( SiOH, SiO^- \) and \( SiOH_2^+ \), where the amphoteric site \( SiOH \) could transform into the other two types under the effect of hydrogen or hydroxyl ions:

\[
SiO^- + OH^- \rightleftharpoons SiOH \rightleftharpoons SiOH_2^+ + H^+
\] (2.2)

Since the surface charge applied on the insulator is the sum of the positive and negative sites (\( SiOH_2^+ \) and \( SiO^- \)), the chemical concentration can be expressed in the form of charges accumulated, which can be electrically sensed. Because of the self-ionization of water, the
concentration of hydroxyl is in the reciprocal relationship with that of hydrogen and the equilibrium constant of the above reactions can be expressed in terms of hydrogen ion only:

\[
\begin{align*}
K_+ &= \frac{[\text{SiOH}^+]_s}{[\text{SiOH}][H^+]_s}, \\
K_- &= \frac{[\text{SiO}^-][H^+]_s}{[\text{SiOH}]}.
\end{align*}
\]  

(2.3)

where the square brackets stand for the concentration of the ions and thus \([H^+]_s\) is the concentration of hydrogen ions present at the interface between the insulator and electrolyte.

To explore the relationship between the concentrations of hydrogen ions at the surface and that in the bulk solution, the Helmholtz & Gouy-Chapman capacitive double layers model is applied.

### 2.7.2 Helmholtz & Gouy-Chapman model

Assuming we apply a positive potential on the reference electrode and we take the hydrogen ions as an example, the double layer model is shown in figure 2.11. The Helmholtz plane can be further divided into inner Helmholtz plane (IHP) and outer Helmholtz plane (OHP), the former one is filled with ions attracted by surface binding site and the latter one represents the locus of the closest hydrated ions towards the solid, whose approach to the solid is limited by the water molecules bound. The Helmholtz plane, composed of both IHP and OHP, results in an almost linear potential decline within the plane, modelling the first capacitance:

\[
C_{Helm} = \frac{A\epsilon_0\epsilon_r}{d_{OHP}}
\]

(2.4)

where \(A\) is the surface area of the interface, \(\epsilon_0\) is the permittivity of free space, \(\epsilon_r\) the relative permittivity of electrolyte and \(d_{OHP}\) the distance from the OHP to solid.
Figure 2.11: Illustration showing (a) SiO$_2$-electrolyte interface in a ISFET, (b) the ionic distribution forming the Helmholtz and Gouy-Chapman layers, (c) associated potential changes and (d) equivalent capacitor

The Gouy-Chapman layer is a diffusion layer from OHP towards the bulk solution following the Boltzmann distribution. Because of the non-linear potential change, Gouy-Chapman capacitance is usually modelled in differential form [48]:

$$\delta C_{Gouy} = \frac{\varepsilon_0 \varepsilon_r}{L_D} \cosh \left( \frac{2 \Psi_G}{2V_t} \right), \quad L_D = \sqrt{\frac{\varepsilon_0 \varepsilon_r V_t}{2n^0 z^2 q}}$$  \hspace{1cm} (2.5)
The first term \( \frac{\varrho_{0}}{L_{D}} \) is simply the unit capacitance of two plates with distance of \( L_{D} \) and the second hyperbolic cosine term describes the effects of mobile charges, where \( z \) is the valence of ions, \( \Psi_{C} \) the potential over the diffuse layer and \( V_{t} \) the thermal voltage. The Debye length \( L_{D} \) is, however, not constant since it is mainly determined by the bulk number concentration of ions \( n^{0} \) (ions/litre).

The voltage on the semiconductor surface would be the potential drop across two capacitive layer from the reference electrode \( \Psi_{ref} - \Psi_{helm} - \Psi_{diff} \). To express the total potential drop \( \Psi_{0} = \Psi_{helm} + \Psi_{diff} \) in terms of pH value, work [49] simplifies the equation into:

\[
\Psi_{0} = 2.3 \frac{kT}{q} \alpha (pH_{pzc} - pH), \quad \alpha = \left( \frac{2.3kTC_{l}}{q^{2} \beta_{int} + 1} \right)^{-1}
\]

where the \( pH_{pzc} \) is a reference pH value which neutralises the charges on the oxide surface, and \( \alpha \) stands for a scaling factor ranging from 0 to 1 describing the reduction of sensitivity from the ideal Nernstian response. In the constituent of \( \alpha \), \( C_{l} \) is the total capacitance of the double layer and \( \beta_{int} \) is the intrinsic buffer capacity of the oxide surface, representing the sensitivity of binding sites in relation to pH changes \( \beta_{int} = \frac{\delta([\text{SiO}^{2-}] + [\text{SiOH}^{+}])}{\delta pH} \).

In the application of DNA sequencing, the emphasis is on relative pH change rather than the absolute pH value, leading to a loose accuracy requirement of solving this parameters. [50] is believed to be the first report of detecting DNA, more specifically SNPs, by monitoring the pH change during nucleotide incorporation using semiconductor.

### 2.8 Extended-gate ISFET & Practical Modelling

The desire to further integrate chemical sensors with signal processing circuits drives people to fabricate the ISFET in an unmodified CMOS process. This is realised by using the extended-gate ISFET approach, first proposed in [51-53], which utilises the fact that the intrinsic passivation layer in the CMOS process is also ion sensitive. [43] has summarised ISFET works in array forms over the last decade, shown in table 2.2. It is interesting to note that the majority of them applied the extended-gate ISFET implemented in unmodified CMOS technology.

One practical ISFET implementation in unmodified CMOS technology is shown in figure 2.12(a). The gate of a buried transistor is extended all the way up to the top metal, which senses the binding charges at the interface between the passivation layer and electrolyte. Compared to the conventional ISFET implementation, sensing principle is the same.
Table 2.2: Comparison of CMOS ISFET arrays over the last decade

<table>
<thead>
<tr>
<th>Year</th>
<th>Tech. ($\mu m$)</th>
<th>Array Size</th>
<th>Pixel Size ($\mu m$)</th>
<th>Application</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>1.2</td>
<td>6 x1</td>
<td>NA</td>
<td>blood gas</td>
<td>[54]</td>
</tr>
<tr>
<td>2004</td>
<td>0.35</td>
<td>2 x2</td>
<td>2.9 x 6.2</td>
<td>cell culturing</td>
<td>[55]</td>
</tr>
<tr>
<td>2005</td>
<td>0.35</td>
<td>16 x 16</td>
<td>12.8 x 12.8</td>
<td>extracellular imaging</td>
<td>[56]</td>
</tr>
<tr>
<td>2006</td>
<td>0.18</td>
<td>5 x 10</td>
<td>50 x 50</td>
<td>DNA detection</td>
<td>[57]</td>
</tr>
<tr>
<td>2008</td>
<td>0.35</td>
<td>3 x 11</td>
<td>57.5 x 57.5</td>
<td>electrolyte imaging</td>
<td>[58]</td>
</tr>
<tr>
<td>2008</td>
<td>0.35</td>
<td>16 x 16</td>
<td>12.8 x 12.8</td>
<td>extracellular imaging</td>
<td>[59]</td>
</tr>
<tr>
<td>2008</td>
<td>0.35</td>
<td>2 x 2</td>
<td>12.8 x 12.8</td>
<td>electrolyte imaging</td>
<td>[60]</td>
</tr>
<tr>
<td>2009</td>
<td>0.18</td>
<td>8 x 8</td>
<td>20 x 2</td>
<td>handheld inst.</td>
<td>[61]</td>
</tr>
<tr>
<td>2009</td>
<td>1.2</td>
<td>16 x 16</td>
<td>105.3 x 81.4</td>
<td>DNA detection</td>
<td>[62]</td>
</tr>
<tr>
<td>2010</td>
<td>0.35</td>
<td>40 x 1</td>
<td>NA</td>
<td>DNA detection</td>
<td>[63]</td>
</tr>
<tr>
<td>2010</td>
<td>0.35</td>
<td>8 x 1</td>
<td>30 x 100</td>
<td>NA</td>
<td>[64]</td>
</tr>
<tr>
<td>2011</td>
<td>0.35</td>
<td>8 x 8</td>
<td>50 x 50</td>
<td>opto-chemical imaging</td>
<td>[65]</td>
</tr>
<tr>
<td>2011</td>
<td>0.35</td>
<td>1.5M</td>
<td>5.1 x 5.1</td>
<td>DNA detection</td>
<td>[30]</td>
</tr>
<tr>
<td>2012</td>
<td>0.18</td>
<td>3 x 3</td>
<td>10 x 10</td>
<td>DNA detection</td>
<td>[66]</td>
</tr>
<tr>
<td>2012</td>
<td>0.35</td>
<td>16 x 16</td>
<td>14 x 14</td>
<td>electrolyte imaging</td>
<td>[67]</td>
</tr>
<tr>
<td>2012</td>
<td>0.35</td>
<td>64 x 64</td>
<td>10.2 x 10.2</td>
<td>electrolyte imaging</td>
<td>[68]</td>
</tr>
<tr>
<td>2013</td>
<td>0.35</td>
<td>40 x 1</td>
<td>104 x 34</td>
<td>DNA amplification</td>
<td>[69]</td>
</tr>
</tbody>
</table>

†: works do not use unmodified ISFET

Figure 2.12: Illustration of an ISFET in an unmodified CMOS technology and its macro-model
but through capacitive coupling over the passivation layer, and hence one more capacitance $C_{\text{pass}}$ is added in the model, shown in figure 2.12(b).

Due to the scaling of the potential drop caused by capacitive division between the chemical and passivation capacitors, it is worthwhile to investigate the influence of $C_{\text{pass}}$. According to the design guide in [48], the Helmholtz capacitance $C_{\text{Helm}}$ can be roughly estimated as $0.14pF/\mu m^2$. The Gouy-Chapman capacitance $C_{\text{Gouy}}$ as we mentioned above depends on the solution concentration and applied voltage on the reference electrode. When there is zero voltage applied, $C_{\text{Gouy}}$ will be in the same order of magnitude as $C_{\text{Helm}}$ in most solutions. Due to the hyperbolic cosine term in equation 2.5, however, applying voltage $\Psi_G$ will make the value of $C_{\text{Gouy}}$ far larger than that of $C_{\text{Helm}}$, resulting in a negligible contribution in a serial connection. On the other hand, the capacitance generated by the passivation layer is extremely small, in the order of $20aF/\mu m^2$ (intrinsic CMOS technology), roughly 4 orders smaller than the chemical capacitance. As a result, the additional capacitance $C_{\text{pass}}$ should have no influence to the charge distribution or potential drop at the insulator surface.

If we define the potential drop in the electrolyte from the reference electrode to the insulator surface as $V_{\text{chem}}$, we have [70]:

$$V_{\text{chem}} = E_{\text{ref}} - (\varphi_{eo} - \chi_{eo}) - \frac{\Phi_m}{q}$$

where $E_{\text{ref}}$ is the potential of the reference electrode, $\varphi_{eo}$ the potential across the electrolyte-insulator interface, $\chi_{eo}$ the electrolyte-insulator dipole potential and $\Phi_m$ the metal work function. Since there is only one pH-dependent parameter $\varphi_{eo}$, modelled by the site-binding theory and double-layer capacitance as described before, we could integrate all the non-chemical related term into one terms $\gamma$ and obtain the following equation [70]:

$$V_{\text{chem}} = \gamma + \frac{2.3\alpha kT}{q} pH$$

Recall from the equation 2.6 that coefficient $\alpha$ varies according to the total capacitance $C_t = \frac{C_{\text{Helm}} C_{\text{Gouy}}}{C_{\text{Helm}} + C_{\text{Gouy}}}$ and $C_{\text{Helm}}$ is independent of solution concentration, we could feasibly drive a voltage on the reference electrode and make $C_{\text{Helm}}$ dominant as we described above, which practically results in a constant $\alpha$ even when pH value changes. Then we could derive the voltage exhibiting on the transistor gate as:

$$V_g = V_{\text{ref}} - V_{\text{chem}} - V_{tc}$$

where $V_{tc}$ is the trapped charge in passivation layer.
As a result, our pH reaction will have a linear mapping on the gate of transistor, and can be processed electronically.

2.9 ISFET Non-ideal effects

ISFETs fabricated in CMOS, however, experience similar challenges to the floating-gate MOSFETs from which they are made, both suffering from the trapped charge induced DC offset and attenuation of the input due to capacitive division of the transconductance by the floating gate capacitor, the effects of which are shown in figure 2.13. As reported in [71], a threshold voltage variation of $-1.32\,\text{V}$ due to trapped charge and a loss of half transconductance efficiency ($g_m/I_D$) were observed. Additionally they suffer from severe drift due to the non-ideal nature of the passivation which also degrades the sensor performance [72]. This is especially problematic when long-term measurements are required as it has been reported to vary from $1.5\,\text{mV}$ to $8.5\,\text{mV}$ per hour in CMOS [73]. All these non-ideal characteristics which exist can be difficult to compensate because of their random and inconsistent nature. Recall the summary of ISFET works in the last decade, table 2.2, different approaches were applied to address these problems. The most straightforward method is to use a high resolution ADC (>$10$ bits) to record everything and process it digitally [63], which demands large data bandwidth in array applications. For removing trapped charge in the device, ultraviolet (UV) radiation has been widely used with biased bulk-substrate voltages to change the conduction and valence band levels, but this requires an external source for

![Figure 2.13: Non-ideal effects present in CMOS based ISFETs [74]](image-url)
compensation [55] [60].

2.10 ISFET front-end circuits

In this section, existing ISFET front-ends are briefly described.

2.10.1 Classical front-ends

As mentioned in the first chapter, along with the process reduction the passivation capacitance becomes so small that it is comparable to the parasitic gate-drain capacitance of transistors, leading to a sensitivity loss due to capacitive division. To compensate for this type of loss in transconductance, a constant-voltage-constant-current (CVCC) readout has been used to bootstrap the capacitor [75, 76], as shown in figure 2.14.

![Figure 2.14: The schematic of a constant voltage constant current front-end](image)

Since the current flowing through the ISFET is fixed by the transistor Mp1, any voltage changes at the gate will cause a same change at the drain via the feedback loop. As a result, the overdrive voltage of ISFET keeps constant. On the other hand, the voltage drop between drain and source is also fixed by a current flowing through a resistor R. Hence there is no variation of voltage difference between any terminals of the ISFET, making it immune to the capacitive division. This is however at the expense of using additional buffers.
2.10.2 Current mode techniques

The Hydrogen Cell concept was proposed in [77] and is shown in figure 2.15. It utilises the logarithmic features of transistors in weak inversion to perform a translinear loop. In addition to this, since the pH value is also in a logarithmic relationship with hydrogen concentration, it is found that the output current is linear to the hydrogen concentration as:

$$I_{out} = I_b \exp\left(\frac{-2V_{ref}}{nU_T}\right)K_{chem}^2[H^+]$$  \hspace{1cm} (2.10)

2.10.3 Programmable gate techniques

$$V_{ref,c}$$  \hspace{1cm} $V_{ctrl}$  \hspace{1cm} $C_p$  \hspace{1cm} ISFET  \hspace{1cm} $C_1$  \hspace{1cm} $I_d$

Figure 2.15: The schematic of a current mode ISFET front-end

Figure 2.16: The schematic of programmable gate ISFET front-end
Work [64, 65] applied the programmable gate techniques to compensate for the trapped charges, as shown in figure 2.16. Intuitively the DC output point is affected by the control voltage $V_{ctrl}$ through a capacitor $C_1$. This approach requires bulky auxiliary circuits and control logic to adjust the programmable gate. Another limitation of this approach is the trade-off between the control range and sensitivity loss due to this additional capacitance. Assuming in the best scenario the control voltage ranges from rail to rail, the compensation range is scaled down to $\frac{C_1}{C_1 + C_p}V_{dd}$ whereas the sensitivity is reduced by a factor of $\frac{C_p}{C_1 + C_p}$. Hence the PG-ISFET solution is limited to specific applications which can tolerate this reduction in sensitivity.

### 2.10.4 pH-to-time techniques

One special application of the programmable-gate techniques is the pH-to-time conversion. Work [66] directly converts the pH variation into the duty cycle of a square wave. The schematic is shown in figure 2.17.

![Figure 2.17: The schematic of the pH-to-time front-end](image)

The ISFET and another NMOS transistor together work like a chemical inverter. A triangle wave is driven on the control voltage of the NMOS transistor. The pH variation in chemical environment will change the switching point of this inverter and hence varies the duty cycle of the output square wave.
2.10.5 Differential readout

To reduce the effects of chemical drift due to reactions happening on the passivation surface, research has been conducted to explore differential techniques which have the potential to minimise this effect. Work [78] integrated a differential structure into a Gilbert Cell readout as shown in 2.18. However, the drift speed for different sensors can be inconsistent, which means this method could not complete solve the drift problem.

![Image of ISFET based Gilbert Cell](image)

Figure 2.18: The schematic of ISFET based Gilbert Cell

2.11 Desirable features of ISFET front-end designs

Considering the signal pattern in semiconductor based DNA sequencing, the required specification for efficient DNA detection is listed below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>minimal</td>
</tr>
<tr>
<td>Noise</td>
<td>low</td>
</tr>
<tr>
<td>Drift</td>
<td>compensated</td>
</tr>
<tr>
<td>Sensitive</td>
<td>High</td>
</tr>
<tr>
<td>Sensitive loss</td>
<td>minimal</td>
</tr>
<tr>
<td>Sensitive mismatch</td>
<td>low</td>
</tr>
</tbody>
</table>

In the following chapter, we will investigate some novel, compact and low-power ISFET
front-end readout circuits which through direct capacitive feedback, are capable of compensating issues of trapped charge, pixel offset voltage, and capacitive division of the input signal, which until now have made ISFET application in sensor arrays challenging.
Bibliography


[22] M. Fedurco et al., “BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies,” *Nucleic Acids Research*, vol. 34, no. 3, 2006,


[31] K. D. Barbee and X. Huang, “Magnetic assembly of high-density DNA arrays for genomic analyses,” *Analytical Chemistry*, vol. 80, no. 6, pp. 2149–2154, 2008,


[57] A. Hassibi and T. H. Lee, “A programmable 0.18-??m CMOS electrochemical sensor microarray for biomolecular detection,” *IEEE Sensors Journal*, vol. 6, no. 6, pp. 1380–1388, 2006,


[60] ——, “Matching the Transconductance Characteristics of CMOS ISFET Arrays by Removing Trapped Charge,” pp. 1074-1079, 2008,

[61] W. Chan et al., “64 pH-ISFET averaging array employing global negative current feedback,” p. 536, 2009,


Chapter 3

ISFET interfaces based on capacitive feedback

3.1 Introduction

In the last chapter we introduced the DNA sequencing techniques using semiconductors and followed on to detail the mechanism and existing problems in current ISFET technologies. The impact of these problems on a chemical reaction is shown in figure 3.1 where the blue line shows the desired ideal signal pattern and the red line shows the deteriorated pattern caused by non-ideal effects. Optimising the performance of the front-end will not only give us better detection results, but also alleviate back-end readout and processing complexity. As mentioned before, large threshold voltage variation (−1.32 V [1]) and drift (1.5 mV to 8.5 mV per hour [2]) will require very large dynamic range and high resolution ADC to handle the output data; and the sensitivity loss (up to 50% [1]) will directly impact the detection resolution. Therefore there is an incentive to improve the detection capability by solving these problems.

In this chapter, a novel methodology is introduced to create compact and low-power ISFET front-end readout circuits, which through direct capacitive feedback are capable of compensating issues of trapped charge, pixel offset voltage, and capacitive division of the input signal. The application of the proposed interface is for reaction monitoring and tracking changes in pH to confirm positive reactions in ISFET arrays. Further exploration of an implementation which simplifies the aforementioned structure by using one transistor as the amplifier in the feedback loop is presented. Finally, a front-end where the feedback
switch is omitted to achieve extreme small area is discussed. A summary and comparison of these three structure is given at the end of this chapter.

3.2 A novel ISFET readout circuit

As mentioned in the Background chapter, different approaches are applied to address non-ideal effects in current ISFET implementations. Apart from the methods using external means, such as UV radiation [3] [4], transistor-level approaches can also be applied such as using a high resolution ADC [5], applying Programmable-Gate [4, 6, 7], and implementing CVCC (constant-voltage-constant-current) structure [8, 9]. All these approaches however require large peripheral circuits overhead. A compact ISFET readout circuit using capacitive feedback will be introduced which can compensate for all ISFET non-idealities.

3.2.1 System Analysis

The functionality and merits of capacitive feedback shall now be described below. Figure 3.2 shows the structure of the proposed interface, where $MP_0$ is the ISFET, $C_p$ is its passivation capacitor and $C_f$ is the feedback capacitor. Capacitor $C_p$ comprises all the coupling capacitors from gate to ground or power supply. The switch $S_0$ is a low-leakage switch which allows the system to have resetting and sensing phases. By adopting this switch structure,
an average leakage current of 45 aA can be achieved [10]. In this switch, the drain-bulk reverse-biased leakage current is minimised by driving the bulk to be the same potential with drain. Dummy transistors are added to cancel the charge injection. Minor delays are also on purposely introduced to ensure the correct charge distribution path.

Figure 3.2: Schematic of the capacitive feedback ISFET interface

The principle of operation is that the gate voltage of the ISFET transistor $MP_0$, $V_g$, is always held at a certain value which makes the drain voltage of $MP_0$, $V_d$, virtual ground to the negative input of the OTA through feedback. As a result when the switch $S_0$ is closed, which is the resetting phase, the output signal is always fixed to a specific value regardless of the value of the chemical reference $V_{ref,c}$ and the amount of trapped charge on the passivation capacitor; in the sensing phase however, $S_0$ is opened and the feedback is now established through the capacitor $C_f$. Hence, to lock the gate voltage according to the capacitive weighting between the passivation and feedback capacitors, the output voltage will compensate the voltage variation due to a pH change. This results in the output voltage $V_{out}$ always inversely tracking the chemical change from a fixed reference. Assuming the chemically dependant voltage $V_{chem}$ to be the input, the DC gain due to the capacitive
weighting can be derived as \( A_{dc} = \frac{C_p}{C_f} \), hence the output voltage will be:

\[
V_{out} = A_{dc} \cdot V_{chem} = \frac{C_p}{C_f} \cdot V_{chem}
\]  

\( \Rightarrow \Delta V_{out} = \frac{C_p}{C_f} \frac{2.3\alpha kT}{q} \Delta pH \)  

Additionally, because all of the four terminals of the ISFET transistor \( MP_0 \) are locked to fixed values, it works in a similar fashion to CVCC (Constant-Voltage-Constant-Current) readout where parasitic gate-source capacitor \( C_{gs} \) is bootstrapped, making it immune to any capacitive division which affects the transconductance.

### 3.2.2 Simulated System Operation

![Figure 3.3: Transient response of the system in two different phases](image)

The circuits functionality during chemical reaction monitoring is depicted in figure 3.3. We used retrospective data from a pH reaction acquired from two ISFET sensors, separated by a DC offset of 20 mV due to trapped charge. Shown is the output response of two readouts interfaced to these two ISFETs. For the resetting phase (\( \phi_1 \), S0 turned on) we see that both
outputs are locked to a fixed potential reference of 1.75 V. Then during the sampling phase ($\phi_2$, $S0$ turned off), both ISFETs successfully track the chemical reaction, immune to the effects of any trapped charges and pixel offset. Furthermore, by using this switching scheme we can guarantee the output will always begin from a fixed reference, which is set during phase $\phi_1$ indicated as the reset zone in figure 3.3, avoiding issues that arise due to ISFET voltage drift.

This approach provides a difference in value from the previous sample rather than an absolute value, which has advantages in allowing the system to reset to a fixed potential and compensating for drift. The reference value for measurements is therefore always the starting pH value, and from this the total change in pH can be deduced. This makes it ideal for monitoring chemical reaction changes rather than absolute values which is suitable for applications involving DNA hybridisation. The absolute pH however can always be deduced if needed as the initial pH of the sample is always known and set to the ideal conditions for the DNA reaction to take place [11].

3.2.3 Stability analysis

The first thing to note in this circuit is that we added a buffer after the OTA to drive the feedback capacitor. It does not only simplify the stability analysis due to its uni-directional signal path, but also increases the drivability for potential scaling of the ISFET array to a larger system, which brings large load capacitance to the output stage. In the resetting phase, the circuit forms a well known simple two-stage amplifier, so we proceeded to analyse the transfer function of the system in the sensing phase only.

Although the real input of the system is at the floating gate, we shall break up the feedback loop at the drain of ISFET transistor $M_{P0}$ to analyse the system’s stability, using the model shown in figure 3.4 where $g_{m1}$ is the transconductance of the feedback OTA and $g_{m2}$ is the transconductance of the ISFET. $Y$ represents the electric admittance of all components, with $Y_f$, $Y_p$ and $Y_C$, representing the feedback capacitor $C_f$, passivation capacitor $C_p$ and the compensation capacitor $C_C$ and resistor $R_C$ connected in series respectively. Additionally, $Y_{L1}$ and $Y_{L2}$ represent the parallel combination of the output impedance and capacitive load of the OTA and ISFET respectively. The transfer function of the circuit is given by:
\[ V_{\text{out}} = \frac{g_{m1}(Y_c - g_{m2}Y_f)}{V_{\text{in}}(V_d)} \]

\[ \frac{g_{m1}(Y_c - g_{m2}Y_f)}{Y_cY_{L2} + Y_cY_{L1} + Y_{L1}Y_{L2} + Y_c9g_{m2}Y_f} \]

Figure 3.4: Open-loop model for stability analysis

\[
V_{\text{out}} = \frac{g_{m1}(Y_c - g_{m2}Y_f)}{V_{\text{in}}(V_d)} = \frac{g_{m1}(Y_c - g_{m2}Y_f)}{Y_cY_{L2} + Y_cY_{L1} + Y_{L1}Y_{L2} + Y_c9g_{m2}Y_f}
\]

and by considering \( G_{L1} = 1/R_{L1} \), and the gain of the second stage \( A_2 = g_{m2} \times R_{L2} \), indicates the existence of poles and zeros in the system at:

\[
p_1 = -\frac{G_{L1}C_f + C_p}{A_2C_c} \frac{C_f}{Y_c}
\]

\[
p_2 = -\frac{C_c9g_{m2}}{C_cC_{L1} + C_cC_{L2} + C_{L1}C_{L2}} \frac{C_f}{C_f + C_p}
\]

\[
z_1 = \frac{1}{C_c(\frac{1}{g_{m2}} + \frac{C_f + C_p}{C_f} - R_c)}
\]

Recalling that the dominant pole of a two stage amplifier with compensation capacitor is \( -\frac{G_{L1}}{A_2C_c} \), when compared with our equation, we see that the dominant pole is increased by a ratio of \( \frac{C_f + C_p}{C_f} \). At the same time however, the DC gain is attenuated by the same factor of \( \frac{C_f + C_p}{C_f} \), resulting in the gain-bandwidth-product to be unchanged.

Furthermore, the second pole is decreased by the same ratio while the zero could be either positive or negative depending on the overall sum of the denominator of \( z_1 \). As there is a sudden introduction of this ratio on the poles and zeros after switching from the resetting to the sensing phase, compensation of the zero for both phases cannot be achieved. It is therefore desirable to give more attention to cancelling out the decreased non-dominant pole
in the sensing phase. This can be accomplished by making the influence of the ratio factor negligible, which can be achieved by making $C_f$ dominant.

### 3.2.4 Noise analysis

To describe the noise performance of the system, we focus on the variation of electronic noise due to the addition of the feedback capacitor. The equivalent transformation for the noise analysis is shown in figure 3.5, where the amplifier $A$ symbolises the whole system excluding the feedback capacitor $C_f$, with $V_{n,a}^2$ being the equivalent input referred noise of amplifier $A$ as it would appear on the floating gate of the ISFET device. $V_{n,i}^2$ is the equivalent input referred noise of the system considering the two capacitors, $C_p$ and $C_f$. Since capacitors do not contribute to any noise, the majority of the amplifier noise $V_{n,a}^2$, as defined by the noise figure, is dominated by the first stage, which in this case is the ISFET stage. The input referred noise contribution can therefore be derived as:

$$V_{n,o}^2 = V_{n,a}^2 \left( \frac{C_p + C_f + C_g}{C_f} \right)^2 = V_{n,i}^2 \left( \frac{C_p}{C_f} \right)^2$$

$$\Rightarrow V_{n,o}^2 = \left( \frac{C_p + C_f + C_g}{C_p} \right) V_{n,a}^2$$

As can be seen, to optimise the noise performance, we need to make the passivation capacitor dominant, and also decrease the coupling capacitor at the gate $G_g$.

![Figure 3.5: Schematic for analysing noise equivalence](image)

#### 3.2.5 Leakage analysis

With an ideal switch, the output signal should respond to the input signal scaled by a ratio of $-C_p/C_f$. In reality, however, the leakage current from the gate node will change the value
of the floating gate voltage, $V_g$ and consequently cause deviation in the output voltage, $V_{out}$. The change of gate voltage due to the leakage can thus be represented as:

$$\Delta V_{g,\text{leak}} = \frac{\Delta Q}{C_{\text{tot}}} = \frac{I_{\text{leak}} \cdot \Delta t}{C_{\text{tot}}}$$  \hspace{1cm} (3.7)

and the corresponding output deviation as a result of this is:

$$\Delta V_{\text{out,leak}} = \frac{C_{\text{tot}}}{C_f} \Delta V_{g,\text{leak}} = \frac{I_{\text{leak}}}{C_f} \Delta t$$  \hspace{1cm} (3.8)

where $C_{\text{tot}}$ stands for the total capacitance seen from the gate node. We note that the amplitude of the output signal also depends on the feedback capacitor, $C_f$. So when measuring chemical changes which are medium-to-long-term (a few seconds), the minimum detectable change is determined by the leakage current when the passivation capacitor is fixed. Quantified as the signal-to-(leakage)drift ratio (SDR) below, this is maximised by keeping leakage to a minimum when the reaction is slow:

$$SDR = \frac{\Delta V_{\text{out,sig}}}{\Delta V_{\text{out,leak}}} = \frac{\Delta V_{\text{in,sig}} \cdot A_{\text{dc}}}{\Delta V_{\text{out,leak}}} = \frac{C_p}{I_{\text{leak}}} \frac{\Delta V_{\text{in,sig}}}{\Delta t}$$  \hspace{1cm} (3.9)

**Low leakage switch**

We apply the low leakage switch introduced in [10], shown in the block figure 3.2, to prevent large drift of the output voltage when the switch is opened during the a pH reaction. Where $\phi_d$, $\overline{\phi}$ and $\overline{\phi}_d$ stand for delayed, inverted and inverted-and-delayed clocks respectively.

---

![Figure 3.6: Low-leakage low-charge-injection switch](image-url)

Terminal $T_1$ connects to the storage node, which is the gate of $MP_0$, while terminal $T_2$ is driven by the output. Transistor $M_1$ forms the core of the switch and $M_2$ cancels the
injected charge. The key purpose of this structure is to set the reference voltage $V_{\text{ref}}$ to be the same as the storage voltage at $T_1$ to minimize the significant drain/source-to-bulk diode leakage. As shown in figure 3.2, the gate voltage of transistor $M_{N0}$ is determined by the bias current and its transistor size, which should be the same as the value generated by the dummy transistor $M_{N0,d}$ with the same bias current, so we can easily apply the same reference to the switch and minimize the leakage.

All the above theory is built on the assumption that the stored voltage will not change after the switch is turned off. The reality, however, is that the total capacitance at the gate is too small to be immune from the charge injection from the switch. The voltage jump caused by the charge injection introduces a constant leakage current leading to a deviation of output voltage. Increasing the gate capacitor $C_g$ will improve the immunity without affecting the gain.

Figure 3.7: Current leakage under light for two different gate capacitors

Figure 3.7 shows the leakage for two different values of gate capacitors: $185\, fF$ and $358\, fF^1$. To measure the leakage effect, we leave the chip under the light which greatly increases the leakage current. 32 front-ends with these capacitors were measured and as can

---

1. This result was measured from 0.35\,\text{um} CMOS process fabrication chip which will be described in later sections.
be seen in figure 3.7, the front-ends with the smaller gate capacitor have approximately 2.5 times more leakage than that with larger gate capacitor. It is therefore preferred to use a larger gate capacitor to minimise leakage.

3.2.6 Design tradeoff when choosing capacitors

As shown in the previous sections, there is a trade off when choosing the sizes of capacitors for the ISFET interface, which can be clearly described with the aid of figure 3.8 to assist the design process. In general, larger capacitors cost more area but can give us a better matching in return. Passivation capacitor $C_p$ should be made as big as possible, subject to area since it improves all other specifications. Choosing a large feedback capacitor $C_f$ has benefits of stability whereas choosing a smaller one improves electronic noise performance and amplitude response. A larger gate coupling capacitor $C_g$ could decrease the effects of charge injection from the switch but would also increase the noise. The choice of the capacitors should therefore be based on the specifications required for a given application. For example, in DNA sequencing, the required detection period is less than 10 seconds but large number of sensor is required, then we could use small size pixel with smaller capacitance values. On the other hand, for the DNA methylation detection, a few sensors to detect around 30 minutes reaction is required, so a possible solution could be applying the front-end with very large passivation capacitance as shown in table 3.1.

![Figure 3.8: Tradeoff in choosing the value of feedback capacitor](image-url)
Table 3.1: Design trade-off for different applications

<table>
<thead>
<tr>
<th>ISFET sensing area</th>
<th>reaction time</th>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>&lt; 10s</td>
<td>DNA sequencing</td>
</tr>
<tr>
<td>large</td>
<td>&lt; 1h</td>
<td>DNA SNP detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Methylation detection</td>
</tr>
</tbody>
</table>

![Microphotograph of fabricated test devices](image)

Figure 3.9: Microphotograph of fabricated test devices

### 3.3 Design and Fabrication

A test chip was designed and fabricated in a commercially available 0.35 μm 2P4M technology provided by AMS. The passivation layer of the given technology is made of Silicon Nitride, $Si_3N_4$, which allows for pH sensing without the need for post-processing. The objective of the fabricated system was to validate the functionality of proposed ISFET interface over a variety of sensing areas and verify its robustness to the non-ideal effects of trapped charge and passivation capacitance as explained previously. To do so the ratio of the passivation capacitor to feedback capacitor was chosen to be close to unity to allow consistency in the output response over all pixel sizes and devices for comparison. The passivation coefficient given by the foundry is 22.65μf/m² which allows an approximate calculation of passivation capacitance, and the necessary feedback capacitor to achieve the desired gain. The transistor sizes for the devices used in the front-end are shown in figure 3.2. The amplifier and buffer used are the standard differential pairs and source follower. The system was optimised for low-power consumption, 848.1 nW for a single front-end with a 3.3 V supply, and minimum area, 60x70 μm². Figure 3.9 shows a microphotograph of the 2x2.5 mm² fabricated device containing ISFET sensors of increasing geometry which interface to the passivation using the top-metal (Metal 4) which forms the passivation capacitor,
Table 3.2: List of fabricated ISFET devices under test

<table>
<thead>
<tr>
<th>Devices</th>
<th>Top metal area ($\mu$m$^2$)</th>
<th>Feedback cap (inc. parasitics) (fF)</th>
<th>Ratio ($\mu$m$^2$/fF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>4624 (68×68)</td>
<td>100 (104.6)</td>
<td>44.2</td>
</tr>
<tr>
<td>D2</td>
<td>3968 (62×64)</td>
<td>90 (94.4)</td>
<td>42.0</td>
</tr>
<tr>
<td>D3</td>
<td>3534 (62×57)</td>
<td>80 (84.06)</td>
<td>42.0</td>
</tr>
<tr>
<td>D4</td>
<td>3100 (62×50)</td>
<td>70 (74.02)</td>
<td>41.9</td>
</tr>
<tr>
<td>D5</td>
<td>2646 (49×54)</td>
<td>60 (63.44)</td>
<td>41.7</td>
</tr>
<tr>
<td>D6</td>
<td>2214 (41×54)</td>
<td>50 (53.86)</td>
<td>41.1</td>
</tr>
<tr>
<td>D7</td>
<td>1770 (30×59)</td>
<td>40 (43.76)</td>
<td>40.1</td>
</tr>
<tr>
<td>D8</td>
<td>1334 (23×58)</td>
<td>30 (33.58)</td>
<td>39.7</td>
</tr>
<tr>
<td>D9</td>
<td>893 (23×39)</td>
<td>20 (23.57)</td>
<td>37.9</td>
</tr>
<tr>
<td>D10</td>
<td>460 (23×20)</td>
<td>10 (13.68)</td>
<td>33.6</td>
</tr>
</tbody>
</table>

$C_p$, whose value is determined by this metal area.

We have fabricated 16 test devices labelled D1-D10 in figure 3.9 with a subset of ten different top metal areas realising different passivation capacitances. Devices under test in this array are listed in Table 3.2, whereby the feedback capacitor $C_f$ has been scaled according to the sensing top-metal area to approximate a similar gain over all devices. Unless the passivation capacitance is very small, e.g. 20 fF, certain slots in top metal are required to be inserted to satisfy the metal stress-relief rules of the given CMOS technology (part of the Design-Rule-Checks), which is required to avoid delamination. Also shown in Table 3.2 is the effective feedback capacitor, which is derived through post-layout simulation. As can be seen, there is around 4 fF of extra capacitance for all the devices due to the coupling of metal routing.

### 3.4 Test setup

In order to test the fabricated device, the bare CMOS dies were wire-bonded which were encapsulated with epoxy on a PCB and then integrated into a micro-fluidic flow cell. The test set-up is shown in figure 3.10. The whole chip is immersed in the solution whose flow is controlled by a syringe connected to the inlet of the chamber. An Ag/AgCl reference electrode is placed at the inlet pipe to bias the devices. Because the chamber is fully sealed except for the inlet and outlet, new solution coming from the inlet will expel any previous solution rather than reacting with it and the solution volume used is large enough to ensure this. As a result, the pH concentration the chip senses is the same as that kept in the syringe
when the chamber is flushed. Tricine 1M KCl pH buffers have been used for the subsequent experimental tests.

![Diagram of experimental setup](image)

(a) Test setup

(b) Photo of experiment setup

Figure 3.10: Experimental Setup

### 3.5 Measured Results

#### 3.5.1 Test of immunity to capacitive division

In this section we show how each device shown in Table 3.2 is immune to gain reduction due to the mentioned decrease of transconductance as a result of capacitive division of the floating gate voltage through the passivation capacitance which exists with ISFETs in CMOS.

Recall the ISFET model we described in previous chapter, figure 3.11, the voltage potential at the floating gate of transistor, $V_g'$, is a linear combination of the reference electrode,
Figure 3.11: Illustration of an ISFET in an unmodified CMOS technology and its macro-model

$V_{\text{ref}}$ and chemical voltages, $V_{\text{chem}}$. Therefore to test the gain of each ISFET interface and prove there is no gain attenuation due to capacitive division, we change the voltage value of the reference electrode to mimic a pH variation and measure the total amplification from a known source. A 100 $mV_{\text{pp}}$, 100 $Hz$ sinusoidal signal is superimposed on the reference voltage, and the output signal is then measured, to derive the gain of the system by taking the ratio of these two signals. Tests were verified in pH 4, 7 and 10 buffer solutions yielding identical results.

The measured amplification capability of all the ISFET DUT (device under test) for 5 different CMOS chips is depicted in figure 3.12. What we see is that the proposed ISFET interface is immune to any capacitive division effects and is able to keep its gain over all ranges of capacitance. We notice that for a single die, the amplification values of devices with the same passivation and feedback capacitance match very well as can be seen from Table 3.3 for the smallest device, achieving a standard deviation of $\sigma=0.0058$ for $n=4$ devices. However the trend in amplification is not constant over all devices which will be justified in the next section due to the value of the passivation capacitance which is not linear with size. We also see that the amplification values for a specific device on different chips varies by about 10%. We think it is might due to the poor control of the passivation layer using unmodified CMOS from die to die, and might come from some experimental errors due to the measuring instruments or miscalibration due to inaccurate pH buffers. This however could be normalised using standard pH calibration. The measured results of mean and standard
deviation of ISFETs with different capacitance are summarised in Table 3.4.

![Figure 3.12: Amplification of ISFETs for 5 different chips](image)

**Table 3.3: Individual Die statistics for \( C_f = 10\) fF**

<table>
<thead>
<tr>
<th></th>
<th>Chip1</th>
<th>Chip2</th>
<th>Chip3</th>
<th>Chip4</th>
<th>Chip5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amp. (n=4)</td>
<td>1.308 ±1.276 ±1.2155 ±1.2625 ±1.2240 ±</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.6m</td>
<td>4.6m</td>
<td>2.5m</td>
<td>7.2m</td>
<td>6.5m</td>
</tr>
</tbody>
</table>

**Non-linear passivation capacitance**

In Table 3.2, the ratio of the top metal area to feedback capacitor ideally should be linearly proportional to the designed gain. Passivation capacitance in ISFETs however is generally poorly defined due to the fact that it is not a conventional two plate capacitor but top metal in contact with an insulator which is in contact with an ionic solution as shown in figure 3.11. We therefore expect the effects of fringe capacitance and the existence of metal slots in the top metal to play a significant role in determining the actual passivation capacitance.

Since the coefficient of passivation capacitance per unit area is unclear, in order to do a relative comparison between devices and deduce the actual expected trend in gain which is measured in figure 3.12, a more detailed analysis of the passivation capacitance is required. To do so, we apply a capacitor model which takes into account the geometry of the top metal to factor in the existing fringe capacitance and metal slots and more accurately estimate the passivation capacitance [12]. The passivation capacitance using this model can be defined as:
where $\epsilon_{\text{eff}}$ is the effective permittivity of double layer passivation, $A$ the area of top metal, $T$ the thickness of top metal, $P$ the perimeter of top metal and $d$ the distance between two plates. The third term in the above equation represents the contribution of the four corners. Using this we can re-calculate the expected passivation capacitance and gain accordingly using the effective feedback capacitance. The results are shown in table 3.4. What we see is that the calculated and measured results of the estimated gain match quite well with an average deviation of 2.74% over all devices, confirming that the fringe capacitance and existence of metal slots play a crucial role in defining the actual passivation capacitance.

Table 3.4: Summary of amplification measurements over all dies

<table>
<thead>
<tr>
<th>Device</th>
<th>Samples</th>
<th>Mean</th>
<th>Standard deviation</th>
<th>Estimated Gain</th>
<th>Deviation(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>5</td>
<td>1.2572</td>
<td>0.0353</td>
<td>1.294</td>
<td>+2.85</td>
</tr>
<tr>
<td>D2</td>
<td>5</td>
<td>1.2880</td>
<td>0.0317</td>
<td>1.242</td>
<td>-3.72</td>
</tr>
<tr>
<td>D3</td>
<td>5</td>
<td>1.3248</td>
<td>0.0307</td>
<td>1.251</td>
<td>-5.92</td>
</tr>
<tr>
<td>D4</td>
<td>5</td>
<td>1.2880</td>
<td>0.0324</td>
<td>1.257</td>
<td>-2.43</td>
</tr>
<tr>
<td>D5</td>
<td>5</td>
<td>1.2844</td>
<td>0.0335</td>
<td>1.264</td>
<td>-1.64</td>
</tr>
<tr>
<td>D6</td>
<td>5</td>
<td>1.2696</td>
<td>0.0319</td>
<td>1.266</td>
<td>-0.33</td>
</tr>
<tr>
<td>D7</td>
<td>5</td>
<td>1.2768</td>
<td>0.0338</td>
<td>1.279</td>
<td>-0.13</td>
</tr>
<tr>
<td>D8</td>
<td>10</td>
<td>1.2564</td>
<td>0.0323</td>
<td>1.232</td>
<td>-1.97</td>
</tr>
<tr>
<td>D9</td>
<td>15</td>
<td>1.2352</td>
<td>0.0303</td>
<td>1.280</td>
<td>+3.53</td>
</tr>
<tr>
<td>D10</td>
<td>20</td>
<td>1.2912</td>
<td>0.0303</td>
<td>1.231</td>
<td>-4.86</td>
</tr>
</tbody>
</table>

3.5.2 Test of pH sensitivity

To test the pH sensitivity of the ISFET sensing front-end, different pH buffers are flowed on top of the device using the flow cell shown in figure 3.10, flushing the chamber with new pH each time to expel the previous solution. Device D1 was used for all chemical experiments. Since the system has the capability to reset the output voltage to its original DC value during the sensing phase, a pH change from this point is always measured, using a buffer of pH 7 as the starting reference. Figure 3.13 shows the step response due to an infused pH change for values 7 to 5, 6, 8 and 9. The change in output voltage for the different pH solutions is summarised in Table 3.5.

A $53.1mV$ change per pH is measured which gives linear sensitivity as shown in the derived calibration curve from the two repeat tests in figure 3.14. Given that the gain
of the front-end is approximately 1.26, an input referred pH sensitivity of 42.1 mV/pH is observed for the silicon nitride in this CMOS process. This is slightly lower than the Nernstian sensitivity ($\alpha=0.7$), but is not unexpected due to the deposition method of the silicon nitride in CMOS [1]. An average accuracy of 91.63% was measured for the two repeats when compared with the linear fit in figure 3.14.

### 3.5.3 Test of long-term drift

To test the drift response of our front-end and the capability to reset the output, thus always initialising a starting point for the system, we conducted a 3 hour experiment in a pH 7 buffer solution. The measured results are shown in figure 3.15, where we can see that the output always starts from the same point after we reset the system and it also drifts at the same rate. The single-ended drift was measured to be 0.21 mV/sec. The effect of this can be regarded as a type of noise and will be analysed in the next section. Resetting the signal also allows us to counteract the effects of chemical drift, which would otherwise cause...
a deviation in output.

Figure 3.14: pH sensitivity curve

Figure 3.15: Long term drift response and the effects of resetting
3.5.4 Noise measurement

As chemical reactions are generally slow in nature, ranging from seconds to minutes, the signals of interest are very low in frequency and thus we only consider noise in the same frequency spectrum. To measure the noise in sub-Hz range, we record the data for over 10 minutes in a constant pH 7 solution and carry out an FFT calculation to get the noise spectrum. The noise test is carried out in wet conditions since chemical noise due to the sensing interface plays an important role. The total noise contributions that were measured are a sum of the chemical, electrical and drift due to leakage:

\[ \text{Noise}_{\text{total}} = \text{Noise}_{\text{elec}} + \text{Noise}_{\text{chem}} + \text{Noise}_{\text{drift}} \]  

The results of the measured noise are shown in figure 3.16, which is mainly dominated by flicker (1/f) noise that is always present at these low frequencies. A relatively large \( K \) value (annotated in the figure) is as expected due to the chemical solution-to-semiconductor interface of the sensors which adds additional chemical noise that is not normally present in dry conditions. It is important to note that the noise calculated is for a differential output which largely alleviates the leakage due to the switches. The coefficient of flicker noise \( K_{\text{chem+leakage}} \) is found to be 1.36\( \mu \)V\(^2\)·Hz. The output integrated noise from 100mHz to 3.5Hz is 5.2mV\(_{\text{RMS}}\), which equivalently gives us 98m pH resolution.

3.5.5 Chemical reaction monitoring

We experimentally validated the system’s capability for reaction monitoring which would emulate a scenario that produces a pH change similar to a DNA base incorporation [11].

First we carry out a pH sensitivity flow cell experiment to measure the transient response of the system. To do so we first start with a buffer solution of pH 7 and then change the pH by infusing different solutions (pH 4, 7, 10) to ramp up and down the pH values, allowing each pH to settle for 20 seconds. The measured results are shown in figure 3.17. What we observed was good sensitivity over multiple repeats and minimum hysteresis when the pH is brought back down to 7.

Following on from this we conducted a diffusion experiment. We submerged the complete ISFET chip in a beaker containing water solution at pH 8. We then manually infuse a high concentration of acid solution and waited for the concentration of the solution to reach equilibrium through the diffusion process. To accelerate the diffusion process we used a
Figure 3.16: Measured output noise power spectrum

Figure 3.17: Transient response to pH variation

magnetic stirrer. The results are shown in figure 3.18. What we see is that the system tracks the reaction, reliably measuring a 2.6 pH change in the time span of 8 seconds.
3.6 Front-end Summary

The achieved performance of the proposed interface with a 100fF feedback capacitance is summarised in Table 3.6. The system achieves amplification, with a measured output of 53.1mV/pH and is capable of monitoring chemical reactions as low as 0.1 pH with a signal-to-noise ratio of 24.67dB over 1 pH change, and a dynamic range of 1.92V which is capable of monitoring over the entire pH range (1-14). Furthermore it has been designed for small-area (60 × 70um²) and low-power operation with an overall power consumption of 848.1nW for a 3.3V supply which is one of the lowest reported. The bias current has been selected to achieve adequate bandwidth due to the high frequency requirement from the gain calibration system which will be introduced in next chapter.

Table 3.7 compares our system with other works reported in the literature. This is the first construct capable of combining compensation for trapped charge, Capacitive division, drift and in-pixel amplification.

\[^{2}\text{Nernst equation can be expressed in terms of concentrations (pH values) only in dilute solutions, valid only for the range from pH 3 to 10 [13]}\]
Table 3.6: Achieved system specification

<table>
<thead>
<tr>
<th>Tech.</th>
<th>AMS-0.35μm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>ISFET Bias Current</td>
<td>100 nA</td>
</tr>
<tr>
<td>Feedback Cap</td>
<td>100fF</td>
</tr>
<tr>
<td>Area</td>
<td>60×70 μm²</td>
</tr>
<tr>
<td>Amplification</td>
<td>1.257</td>
</tr>
<tr>
<td>Power</td>
<td>848.1 nW</td>
</tr>
<tr>
<td>GBP</td>
<td>85.8k</td>
</tr>
<tr>
<td>pH sensitivity</td>
<td>42.1 mV/pH</td>
</tr>
<tr>
<td>Output Noise (0.1-3.5 Hz)</td>
<td>5.2 mVrms</td>
</tr>
<tr>
<td>pH sense resolution (53.1 mV/pH)</td>
<td>0.1pH</td>
</tr>
<tr>
<td>Dynamic Range (1%THD)</td>
<td>1.92 V</td>
</tr>
</tbody>
</table>

Table 3.7: Comparison of ISFET readouts with integrated compensation schemes

<table>
<thead>
<tr>
<th>Year</th>
<th>Tech. (μm)</th>
<th>Size (μm²)</th>
<th>TC comp.</th>
<th>C_pass comp.</th>
<th>drift comp.</th>
<th>Amp.</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>0.35</td>
<td>163</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[4]</td>
</tr>
<tr>
<td>2010</td>
<td>0.35</td>
<td>3000</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[6]</td>
</tr>
<tr>
<td>2011</td>
<td>0.35</td>
<td>2500</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>[7]</td>
</tr>
<tr>
<td>2012</td>
<td>N/A</td>
<td>4-202M</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>[14]</td>
</tr>
<tr>
<td>2013</td>
<td>0.35</td>
<td>460-4624</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>This work</td>
</tr>
</tbody>
</table>

TC comp. - Compensation for trapped charge
C_pass comp. - Compensation for capacitive division
drift comp. - Compensation for drift
Amp. - in-pixel amplification

3.7 Single stage capacitive feedback interface

The main challenge in using the previously proposed front-end in arrays is the infeasible method of multiplexing the pixel. To lock the gate voltage of an ISFET transistor, all the circuits of the front-end have to be turned on, otherwise the gate would be under the manipulation of the reference voltage $V_{ref}$ via the switch, shown in figure 3.19.

To facilitate our front-end for use in large arrays, we have to further reduce the power consumption of a single pixel. A simple reduction in bias current may solve the problem, but would limit the bandwidth. On the other hand, in our chemical detection applications, specifically DNA incorporation monitoring, linearity and output range are not the prioritized concerns, hence we could adopt a single-stage amplifier instead of the two-stage one to largely
lower the power consumption, which will be discussed in the following subsection.

3.7.1 Pixel Schematic

The schematic of the single-stage front-end is shown in figure 3.20. It directly builds the feedback between the gate input and drain output. This structure shares the same two-phase operation scheme with its two-stage counterpart. The output voltage will respond to the chemical input inversely to compensate any voltage in the gate node, also with the gain of $A = \frac{C_p}{C_f}$.

One immediate benefit apart from power saving is the relaxed stability consideration. However, due to the limited gain of a single stage amplifier, the linearity of this front-end is sacrificed. A more concerning problem is the output range.

Unlike the two-stage front-end where the drain voltage is locked, in the one-stage front-end the drain node directly outputs the readout value, which needs to be higher than a certain value to maintain the transistor in saturation. Even in the saturation region, limited $g_{ds}$ will also deteriorate the accuracy. $V_{source}$ is tunable to set up a desired DC output value. The output response is the same as its two-stage counterpart:

$$\Delta V_{out} = \frac{C_p}{C_f} \frac{2.3\alpha kT}{q} \Delta pH$$  \hspace{1cm} (3.12)

3.7.2 Design and Fabrication

A 32x32 array was designed using the same AMS-0.35um 2P4M technology shown in figure 3.21, whose passivation layer is made of Silicon Nitride. Our objective was to investigate
the functionality of the single stage front-end. Top metal (Metal-4, orange material shown in figure 3.21) interfaces to the insulator and forms the passivation capacitor, whose value is determined by the metal area. The size of each sensor pixel is $55 \times 55\mu m^2$, which provides approximately $93\, fF$ capacitance, using parameters from the data-sheet provided by the foundry and our previous analysis. The feedback capacitor is around $72.8\, fF$, indicating an estimated gain of 1.28.
3.7.3 Test of Amplification

Due to the large number of pixels, we did not drive sine waves to each pixel to estimate the amplification. Instead, we exerted a step change on the reference electrode, and recorded the corresponding output change in the whole array. Figure 3.22 shows the gain distribution of the whole array. The mean gain is 1.29 while the standard deviation is 11.7m. The tested gain value matches the theoretic value closely with less than 1.6% error. The 3-σ range covers from approximately 1.24 to 1.33, indicating only a 7.26% sensitivity mismatch, which is much better than expected.

![Figure 3.22: Gain Distribution and its cumulative counts](image)

3.7.4 Test of pH sensitivity

To test the pH response of the ISFETs, we flowed pH 5-9 buffers on top of the die, with a pH 7 solution used to wash out the previous pH buffer whilst providing a chemical reference point. This introduces instant chemical step changes, shown in figure 3.23. After multiple step change measurements, we observed a 40.0mV response for each pH change. Considering the gain of the sensor front-end, whose mean value is 1.26, the original sensitivity is approximately 31.7mV/pH.
3.7.5 Test of Noise

The noise measurements were carried out the same as described before: to measure the noise spectrum in sub-Hz domain, we recorded the data over 100 seconds and used the FFT calculation, shown in figure 3.24. According to the calculation result, the output integrated noise from 100mHz to 3.5Hz is 5.9mVRMS, which equals to 4.5mVRMS input-referred noise.

The value of the noise is slightly larger than the value we got in the two-stage front-end, the main reason is that we only tested the single-ended output in the single-stage front-end, where electronic leakage (0.57mV/s) plays an important role, and leads to a worse noise performance.

3.7.6 Summary of single stage interface

In the last few subsections, we have investigated the performance of a single-stage interface. It is feasible to achieve some benefits such as lower power and smaller area. On the other hand, the single stage amplifier itself will inevitably suffer from lower amplification. The simulated results show the 1% THD range is only 633mVpp whereas the value of its two-
stage counterpart is three times more than that (1.92V). The specification of this single-stage interface is listed in Table 3.8. This structure is the most suitable for array implementation.

### Table 3.8: Specifications of the single-stage interface

<table>
<thead>
<tr>
<th>Tech.</th>
<th>AMS-0.35um</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Bias Current</td>
<td>300 nA</td>
</tr>
<tr>
<td>Feedback Capacitance</td>
<td>≈ 72.8 fF</td>
</tr>
<tr>
<td>Area</td>
<td>55 × 55 um²</td>
</tr>
<tr>
<td>front-end Gain</td>
<td>≈ 1.26</td>
</tr>
<tr>
<td>pH sensitivity</td>
<td>31.7 mV/pH</td>
</tr>
<tr>
<td>Output Noise (0.1-3.5Hz)</td>
<td>5.9 mV&lt;sub&gt;rms&lt;/sub&gt;</td>
</tr>
<tr>
<td>pH sense resolution</td>
<td>0.1 pH</td>
</tr>
<tr>
<td>Dynamic Range (1%THD)</td>
<td>633 mV</td>
</tr>
</tbody>
</table>

3.8 Three-transistor ISFET readout circuit

As shown in the investigation of the single-stage front-end, the main challenge of shrinking the size of sensors is the necessity for capacitors to constrain the electronic leakages from the switch. Hence we explore a simpler structure using the intrinsic capacitance of the device.
as the feedback capacitor consuming only three transistors.

### 3.8.1 Pixel Schematic

The proposed 3-T ISFET readout consists of just one ISFET transistor ($M_{n0}$) biased by constant current as shown in figure 3.25(a), which can be selected by a switch ($M_{n1}$) when used in an array. In the figure, $V_{chem}$ integrates all the pH related chemical potentials [1] and $V_{ref}$ is an external reference voltage applied to the reference electrode to set up the DC operation of the front-end. $C_p$ and $C_{gd}$ stand for the passivation and parasitic gate-drain capacitance respectively, and together determines the electrical gain of the front-end.

![Figure 3.25: Schematic of proposed 3-T front-end and its simplified model](image)

The simplified topology is shown in figure 3.25(b), from which we can see that the close loop gain is defined as $A = C_p/C_{gd}$ provided that the open loop gain is large enough which is the true case especially when the transistor is biased in weak inversion. Again, because the gate terminals of the ISFET transistor $M_{n0}$ is locked via feedback, it works in a similar fashion to CVCC (constant-voltage-constant-current) readout, making it immune to any capacitive division which affects the transconductance. The output response to pH changes should be:

$$\Delta V_{out} = \frac{C_p}{C_{gd}} \frac{2.3\alpha kT}{q} \Delta pH$$

#### 3.8.2 Design and Fabrication

An 8x8 array was designed using an AMS – 0.35um 2P4M technology shown in figure 3.26, whose passivation layer is made of Silicon Nitride. Our objective was to investigate the
functionality of the proposed front-end under extreme small cases. Top metal (Metal-4, orange material shown in figure 3.26) interfaces to the insulator and forms the passivation capacitor, whose value is determined by the metal area. As shown in the figure, the size of each sensor pixel is only $8 \times 8\mu m^2$, which provides approximately $1.7fF$ capacitance, using parameters from the data-sheet provided by the foundry.

![Microphotograph of the fabricated array](image.png)

Figure 3.26: Microphotograph of the fabricated array

### 3.8.3 Test of Amplification

According to the ISFET model, the voltage potential at the gate of the transistor is a combination of reference ($V_{ref}$) and chemical voltages, ($V_{chem}$). Thus to test the gain of each ISFET sensor, we change the voltage value of the reference electrode to mimic the pH variation. Figure 3.27a shows the output response to a reference voltage sweep, and a linear relationship is observed providing a gain of 8.05. The test results also fit the post-layout simulation well, except that its value cannot exceed 2.3V due to the NMOS switch.

To confirm the gain over all devices, we conducted a statistical analysis for 320 pixels over 5 chips. A $100mV_{pp}$ $100Hz$ sinusoidal signal is superimposed on the reference voltage, and the output signals are measured, whose ratio also represents the gain. The results of the mean and standard deviation of the gain over the 5 chips is summarised in table 3.9. A relatively large in-pixel amplification is achieved whose average values are also consistent over chips. However, the gains of different pixels vary a lot due to poor control of the parasitic feedback capacitor.
3.8.4 Test of chemical response

To test the pH response of the ISFETs, we flow pH 5-9 buffers on top of the array, with a pH 7 solution used to wash out the previous pH buffer and provides a chemical reference point. This introduces instant chemical step changes, shown in figure 3.28. A sensed output of about 200mV/pH is observed. Considering the electronic gain of this front-end (around 8), the real pH sensitivity of these devices is about \(25.8\text{mV/pH}\).

3.8.5 Test of noise

As chemical reactions are generally slow and thus low in frequency, we only consider the noise at the same end of the spectrum. To measure the noise spectrum in sub-Hz domain, we recorded the data over 100 seconds and used the FFT calculation, shown in figure 3.29. The noise test is carried out in wet conditions where chemical noise plays a dominant role.
According to the calculation result, the output integrated noise from 100mHz to 3.5Hz is 1.5mVRMS, which equals to 188μVRMS input-referred noise.

3.8.6 Summary of 3T Interface

In the last section, we explored the possibility of utilising the parasitic Drain-Gate capacitor to act as the feedback capacitor for an ISFET front end. The feedback structure locks
the gate voltage of the ISFET, realising a constant-voltage-constant-current structure which avoids the problems of capacitive division. The amplification of the front-end is determined by the ratio of passivation to feedback capacitance, and as a result of the extremely small value of parasitic capacitance, a mean gain of 8 can be realised even with 1.7fF passivation capacitance. However, the poor quality of both the parasitic and passivation capacitance increases the distribution of the measured gain.

Because of its one-transistor structure and low-frequency operation, the interface consumes ultra low power (66nW). The input-referred integrated noise from 0.1Hz to 3.5Hz is only around 180μVrms.

The specification of the proposed front-end is summarised in table 3.10. The pH sense resolution is calculated through the ratio of output pH sensitivity to output noise.

| Table 3.10: Specifications of the proposed interface |
| :----------------- | :----------------- | :----------------- |
| Tech. | AMS-0.35um |
| Supply voltage | 3.3 V |
| Bias Current | 20 nA |
| Drain-Gate Capacitance | ≈ 0.2fF |
| Area | 8× 8 um² |
| front-end Gain | ≈ 8 |
| Power | 66nW |
| pH sensitivity | 25.8mV/pH |
| Output Noise (0.1-3.5Hz) | 1.5mVrms |
| pH sense resolution | > 0.01pH |

3.9 Conclusion

In this chapter first we introduced a low-power, compact sensor interface methodology which overcomes some of the undesirable factors which are present with ISFETs in unmodified CMOS. By applying a feedback to the gate directly from the output terminal, factors such as trapped charge can be removed and drift can be eliminated periodically. The feedback inherently locks the gate voltage of the ISFET, realising a constant-voltage-constant-current structure which bypasses the capacitive division. Thanks to this mechanism, the sensitivity is well maintained even when a parasitic capacitor is used.

Using this methodology we introduced three novel types of front-ends for ISFET sensors,
the two-stage, single-stage and 3-Transistor. Their comparison is listed in table 3.11.

<table>
<thead>
<tr>
<th>Front-end</th>
<th>Area</th>
<th>Power</th>
<th>Leakage</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stage</td>
<td>large</td>
<td>medium</td>
<td>medium</td>
<td>large linearity and output range</td>
<td>relatively power hungry, large area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low power, medium area</td>
<td></td>
</tr>
<tr>
<td>Single-stage</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>small area, low power</td>
<td>bad linearity, large area no reset phase, large gain deviation</td>
</tr>
<tr>
<td>3-Transistor</td>
<td>small</td>
<td>low</td>
<td>low</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the conventional two-stage interface, the design considerations have been investigated, and a tradeoff is derived for gain, stability and noise, whilst the other two do not have these concerns because there is no two-stage feedback connection.

Because of their simple structure and low-frequency requirement, all the front-ends consume low power, especially for the single-stage and 3-T structures, which facilitates implementations for large arrays which has potential use for DNA microarrays. In terms of scalability in future, both two-stage and single-stage front-ends will suffer from the leakage current from switches unless post-processing is applied to replace the intrinsic passivation layer. The 3-T structures could however enjoy the process improvements.

In our front-ends, the gain is determined by the ratio of passivation to feedback capacitance. This can potentially limit our control over the gain accuracy, given the passivation capacitance is determined by uncontrollable fabrication parameters such as thickness and conformality. We have also shown through analysis the effect of fringe capacitance and how slots in the top metal can also play a role in determining the passivation capacitance. Based on these findings, our estimation of passivation capacitance can accurately explain the measured gains in our devices. These aforementioned issues can potentially cause a deviation in amplification for different devices. The problem is more severe in 3-T structure since the feedback is just the parasitic Gate-Drain capacitance. The in-pixel amplification, however, can still be achieved reliably given it can be tuned by changing the ratio of $C_p$ to $C_f$ if required.

The choices of both passivation and feedback capacitors should be ultimately determined by the specification of the application. For example we can keep increasing the top metal area to realise a large passivation capacitor which allows for long-term monitoring, typically a few
minutes, or use a small top metal yielding a compact pixel footprint which can implement larger arrays but would be more suitable for short-term monitoring applications such as DNA sequencing which has a duration of several seconds [15].

In terms of noise, the system is mainly restricted by the electronic leakage from the switch, and hence the better noise performance is observed in the 3-T structure even though it occupies much less area. Optimisation of the leakage could considerably improve the sensing resolution.

Finally, the use of the reset phase allows the opportunity for cancellation of any unwanted offset in the measurement through schemes such as correlated double sampling [16], in addition to measuring from a fixed reference which is ideal for monitoring of a pH change where we expect a certain chemical reaction to occur. Through our novel interface and compact design we have achieved a potentially scalable system that can be applied to a number of sensing applications ranging from the use of simple single or differential pH monitoring for on the spot measurement in point-of-care systems, to medium-scale DNA micro-arrays for DNA hybridisation detection and potentially with further design larger arrays of 10,000 pixels for target specific DNA sequencing.
Bibliography


2013.


[16] Y. Chen and Y. Xu, “Column-Parallel Digital Correlated Multiple Sampling for
2012,
Chapter 4

Automatic Gain Control System for ISFET array compensation

In Chapter 3 we introduced new front-ends with feedback to the ISFET gate to compensate for several non-idealities in these sensors. A limitation identified however was the dependence on both passivation and feedback capacitance mismatch which would cause a deviation in the gain of pixels. This deviation is generally process related determined by the fabrication of the capacitor, whose dimension, thickness and uniformity are difficult to control. Therefore it necessitates a closed loop compensation scheme if it is to be eliminated. In this chapter we design a novel automatic gain control system specifically to deal with non-idealities when implementing ISFET arrays. The concept of gain calibration is shown in figure 4.1.

Figure 4.1: Concept of gain calibration

A high frequency sine wave is superimposed on the chemical signal via the electrode
reference immersed in solution. The amplitude of the sine wave signals after the sensor array carries information of local sensitivity. By calibrating these sine waves to be consistent with amplitudes, the chemical signals detected for all the sensors should experience the same gain and therefore eliminating any deviation in sensor performance.

4.1 System Considerations

ISFET sensors generally suffer from sensitivity mismatch due to the non-uniform passivation layer. For sensitivity calibration, we can either calibrate the whole array and store the information in memory or repeat the adjustment process every time before we sample the sensor value without storage. Clearly, the former method has circuit complexity in terms of data storage and data retrieve time while the latter one requires the sampling time to be extended before reading pH values to account for the settling time. However, considering the possibility of pipelining, for instance calibrating pixel $no.x$ when the pixel $no.(x - 1)$ is under reading process, we may alleviate the constraint on settling time which is the time for calibration. Additionally, real-time calibration is ideal for varying passivation capacitance during the time of use due to chemical reactions occurring on the passivation surface. As a result, a real-time calibration system is adopted. A periodically calibrating scheme however is still useful in the cases where accuracy is not prioritised.

To set up a consistent reference point to calibrate the gain for all the pixels, the reference electrode potential applied in the solution is superimposed with a high-frequency small-amplitude sinusoidal wave. By doing so, the low-frequency sensed chemical information and a high frequency sinusoidal which is artificially applied are combined together. The variation at the high frequency sinusoidal signal then carries information of the sensors’ local sensitivity. The same concept has also been used in [1].

Considering the ISFET gain mismatch range, the proposed AGC needs to be able to calibrate the gain by a factor of 2, e.g. from 70% to 140%. However, less than 1% residual error is targeted to generate less than 1% variation in the gain from sensor to sensor. The settling time is determined by the front-end bandwidth and should be as short as possible to allow fast switching when used in an array. Because of these unique requirements, existing AGC implementations which support more than 60dB tunability but poor accuracy are not applicable for the required system.

The system diagram is shown in figure 4.2, where $V_{ctrl}$ is the gain control signal to finish
Automatic Gain Control System for ISFET array compensation

Figure 4.2: Diagram of the automatic gain control system

Figure 4.3: Flow diagram of the signal path in the AGC

a closed loop. Figure 4.3 represents the flowing graph of the signal path. The input signal from the front-end ISFET contains both low-frequency chemical signal (with subscript $L$) and high-frequency sinusoidal wave (with subscript $H$), which are tuned by a VGA (variable gain amplifier) with same amplification $k_a$. Due to their distinct frequency difference, they can easily be separated by a highpass filter. This is assuming the peak detector is ideal and can measure the peak-to-peak amplitude of input sine wave without introducing any other effect. The amplified peak is compared with an external reference signal $V_{ref}$, and any difference between them will be used to adjust the total gain until they are matched. A
dominate pole (transfer function $H(s)$) is introduced to stabilize the feedback system.

The above described AGC system, however, is an inherently non-linear system where the settling time is input dependent. As a result we should insert two additional blocks (dotted) shown in figure 4.2 to provide an exponential relationship and consequently allow the system to operate linearly in decibel [2]. Then the time constant of the system becomes independent of the input and can be derived as follows:

\[
\begin{align*}
A &= k_a \cdot V_{ctrl} \\
V_t &= \left[ V_{ref} - k_0 \cdot \ln\left( \frac{A_{V_H}}{k_1} \right) \right] \cdot H(s) \\
V_{ctrl} &= k_2 \cdot \exp\left( \frac{V_t}{k_3} \right)
\end{align*}
\]

\[
\Rightarrow V_{ctrl} = k_2 \cdot \exp\left[ \frac{V_{ref} - k_0 \ln\left( \frac{k_1 V_H V_{ctrl}}{k_1} \right)}{k_3} \cdot H(s) \right]
\]

\[
\Rightarrow \ln(V_{ctrl}) = \frac{-k_0 H(s)}{k_3 + k_0 H(s)} \ln(V_H) + \frac{V_{ref} H(s) + k_3 \ln(k_2) + k_0 H(s) \ln(k_1)}{k_3 + k_0 H(s)}
\]

where $k_a, k_0, k_3$ are the coefficients of function blocks shown in figure 4.2.

For a non-variable external reference, the second term is nothing more than a constant. If the transfer function $H(s)$ is set to be an ideal low pass gm-C stage, say $H(s) = \frac{g_m s}{s + C}$, the gain control signal $V_{ctrl}$ will respond to the input linearly in decibel:

\[
\ln(V_{ctrl}) = \frac{-k_0 g_m}{s \cdot k_3 C + k_0 g_m} \cdot \ln(V_H) + K
\]

where $K$ is a grouping of constant terms and the time-constant of this system is given as:

\[
\tau = \frac{k_3 C}{k_0 g_m}
\]

### 4.1.1 Simplified system omitting the log amplifier

Since the proposed system is used to adjust the sensitivity deviation, whose input amplitude level changes by only a small amount, we can even omit the log amplifier before the comparison with the reference [3], shown in figure 4.4. The proof of this is as follows:

\[
\begin{align*}
\left\{ \begin{array}{l}
V_{out,H} = k_a \cdot V_{ctrl} \cdot V_{in,H} \\
V_{ctrl} = k_2 \cdot \exp\left[ \frac{V_{ref} - V_{out,H}}{k_3} \right] \cdot H(s) \end{array} \right.
\end{align*}
\]

\[
\Rightarrow \ln(V_{out,H}) = \ln(k_a) + \ln(k_2) + \ln(V_{in,H}) + \frac{V_{ref}}{k_3} \cdot H(s) - \frac{V_{out,H}}{k_3} \cdot H(s)
\]
If we set the logarithmic input \( \ln(V_{in,H}) \) to be \( x \) and the corresponding output \( \ln(V_{out,H}) \) to be \( y \), we can simplify to equations 4.8 after grouping all the constants into \( K \):

\[
y = x - \frac{H(s)}{k_3} \cdot e^y + K
\] (4.8)

Assuming the initial state has converged through the feedback, which means \( V_{out,H} = V_{ref} \), then a Taylor series of \( e^y \) can be expanded at point \( y = \ln(V_{ref}) \) as:

\[
e^y = e^{\ln(V_{ref})} \cdot (1 + y - \ln(V_{ref}))
\] (4.9)

Replacing the term \( e^y \) in equation (4.8) by the Taylor series shown in (4.9), we have a linear equation as follows:

\[
\left[ \frac{k_3 + H(s)V_{ref}}{k_3} \right] \cdot y = x + K'
\] (4.10)

Again, setting the transfer function \( H(s) \) to be the same lowpass gm-C stage, then the time constant of this simplified system is given by:

\[
\tau_{sim} = \frac{1}{P3db} = \frac{k_3 \cdot C}{g_m \cdot V_{ref}}
\] (4.11)

Comparing it with the expression in (4.5), the simplified time constant is now dependent on an input value \( V_{ref} \) which is no longer a linear system. However, if we force the reference to be unchanged, the total system can still be regarded as a linear-in-dB feedback system.

Since the calibration step is required every time before sensing in the real-time system, it is desired to have a fast tracking system so that large data rate can be realized. Due to this requirement, time constant should be decreased as much as possible. There are however several limitations to achieving this which need to be discussed and analysed in the following sections.
4.1.2 High frequency harmonic noise

The sinusoidal wave amplitude detector, or peak detector, shown in figure 4.2, could be ideally realized through certain mathematical conversions, but this requires either perfect 90° phase shifter [4] or a combination of integrator, differentiator and multiplier [5]. Due to the complex implementation of these, a common alternative of adopting a rectifier followed by a low pass filter is more favourable, shown in figure 4.5.

\[ |\sin(\omega_0 t)| = \frac{2}{\pi} - \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\cos(2k\omega_0 t)}{4k^2 - 1} \] (4.12)

The second order harmonic component (k=1) has a considerable significant value, which will cause an oscillation of the final gain around the desired value. The fluctuation of the final gain is proportional to the control signal \(V_{ctrl}\) whose small signal response also depends on its quiescent condition. Since the fluctuation is a peak-to-peak signal, the final expression needs to be doubled as:

\[ \Delta V_{ctrl,pp} = |V_{in,H}| \cdot k_a \cdot V_{ctrl,dc} \cdot \frac{4}{3\pi} H(\omega_{2nd}) \cdot f'(V_i) \cdot 2 \] (4.13)

where \(\omega_{2nd} = 2\omega_0\) is the radian frequency of second order harmonic and \(f(V_i) = k_2 \exp\left(\frac{V_i}{k_3}\right) = V_{ctrl}\). Because the system is stabilized through a negative feedback, the quiescent control voltage should satisfy the following relationship:

\[ |V_{in,H}| \cdot k_a \cdot V_{ctrl,dc} \cdot \frac{2}{\pi} = V_{ref} \] (4.14)

then the ratio of the signal (DC component) to noise (harmonic) ratio is:

\[ SNR = \frac{V_{ctrl,dc}}{\Delta V_{ctrl}} = \frac{V_{ctrl,dc}}{4V_{ref} H(\omega_{2nd}) f'(V_i)} = \frac{3k_3}{4V_{ref} H(\omega_{2nd})} \] (4.15)
The attenuation is realized by the low pass filter $H(s)$, which equals to $\frac{2m}{s}$, therefore the SNR can be rewritten as:

$$\text{SNR} = \frac{3 \cdot k_3 \cdot C \cdot \omega_{2nd}}{4 \cdot V_{ref} \cdot g_m} \quad (4.16)$$

By replacing parameters in equation 4.11 with SNR, time constant is actually limited by both the frequency of the sinusoidal wave and the SNR required:

$$\tau_{sim} = \frac{4A_{pd} \cdot \text{SNR}}{3 \cdot \omega_{2nd}} \quad (4.17)$$

![Graph showing the relationship between time constant (sec) and harmonic frequency (Hz) for different SNR levels: 40dB, 50dB, 60dB, 70dB, 80dB, 90dB, 100dB. The graph indicates a decrease in time constant as frequency increases for each SNR level.](image)

Figure 4.6: Limitation of time constant due to SNR requirement

This limitation can be visualized in figure 4.6. For example, if the frequency is 100kHz and the required SNR is 40dB, then a time constant of up to $4 \cdot 100 / (3 \cdot 2\pi \cdot 200 \cdot \pi) = 67.5 \mu s$ can be realized. The simulation using ideal function blocks is shown in figure 4.7. The 0.195Np (Natural Logarithmic) fluctuation equals to 0.98% amplification variation.

Note that the rectifier has an attenuation of $\pi/2$ in terms of DC gain compared to an ideal amplitude detector, causing the introduction of a scaling coefficient $A_{pd} = \frac{2}{\pi}$. This new coefficient, however, changes the initial state to $V_{out,II} = V_{ref} / A_{pd}$. Recall the derivation of the time constant of a simplified system from equation (4.6) to equation (4.11), we find $A_{pd}$ can be cancelled and hence plays no role in the trade-off between SNR and time constant.
Besides that, it takes $5\tau$ to achieve 99% accuracy, which bottlenecks the tracking process when we switch multiple pixels as would be the case in an array.

### 4.1.3 Harmonic suppression via additional lowpass filters

The most intuitive way to overcome the SNR limit is to insert another lowpass filter after the rectifier, whose poles are higher than the dominant pole, shown in figure 4.8, where $H_1(s)$ is the old gm-C stage and $H_2(s)$ a lossy unit-gain lowpass filter.
Time constant and attenuation in multi-poles system

Let’s firstly consider the influence on the time constant by adding one additional pole higher than the dominant one. The transfer function of a two pole system is given by:

\[ H_{1,2}(s) = \frac{1}{(1 + s/p_1)(1 + s/p_2)}, \quad p_2 > p_1 \]  \hspace{1cm} (4.18)

and its corresponding step response is:

\[ h(t) = \frac{p_1 p_2}{p_2 - p_1} (e^{-p_1 t} - e^{-p_2 t}) \]  \hspace{1cm} (4.19)

It is obvious that when \( p_2 \gg p_1 \), the second negative term inside the bracket in equation (4.19) \( (e^{-p_2 t}) \) quickly converges to zero and loses its impact, approximating to the equation of a one-pole system. When the two poles are close in value, however, the response is not so intuitive. To analyse this the extreme condition where two poles are located at the same frequency is now discussed. When \( p_1 = p_2 \), the step response is defined as:

\[ y(t) = 1 - e^{-p_1 t} - p_1 t \cdot e^{-p_1 t} \]  \hspace{1cm} (4.20)

Due to the existence of the third term \( (p_1 t \cdot e^{-p_1 t}) \), the new response takes a longer time to track the step change. But because the exponential part \( e^{-p_1 t} \) decays much faster than the linear part \( p_1 t \), the extra time spent is not significant and worth the tradeoff to improve SNR.

Figure 4.9 shows the effect after a second pole is added. The x-axis shows the normalized frequency of the second pole whose number is obtained by \( p_2/p_1 \). The above figure shows the time required to obtain 99% accuracy normalized by the time in the single pole system. As can be seen the time needed is only increased by a factor of 1.5 in the worst case (two repeated poles). In the middle figure, the obvious attenuation effect of the additional pole is indicated. The response frequency is chosen to be 10 times \( p_1 \), as a result 10 times more attenuation can be realized.

In our AGC system, a small time constant and large high-frequency attenuation filter is required. As was analysed previously, however, both the time constant and attenuation is proportional to the single pole in a single-pole system. If we define the Figure Of Merit to be the ratio of attenuation index to time constant, the FOM is inherently fixed unless we change the filter to a multi-poles system, which is shown in the bottom figure.

Another conclusion can be found is that the highest FOM always happens in the case where both poles are at the same frequency.
Optimal condition in dual-pole and tri-pole system

Let’s firstly consider the situation with one additional pole $p_1$. Recall the equation (4.10) we derived in section 4.1.1:

$$y = \frac{k_3}{H(s)V_{ref} + k_3} x + K'$$  \hspace{1cm} (4.21)

where $H(s)$ is the combination of gm-C stage and additional lowpass filter, equals to $\frac{g_m}{sC}\frac{p_1}{s^2 + \frac{g_m}{sC}}$. As a result, the equation can be rewritten as:

$$y = (1 - \frac{g_m \cdot V_{ref} \cdot p_1}{s \cdot k_3 C + s \cdot k_3 C p_1 + g_m p_1 V_{ref}} ) x + K'$$  \hspace{1cm} (4.22)

To avoid the overshoot in step response, the two poles should be real numbers, which requires:

$$p_1 \geq \frac{4g_m V_{ref}}{k_3 \cdot C}$$  \hspace{1cm} (4.23)
Automatic Gain Control System for ISFET array compensation

The best performance is obtained in a critical damping situation, so we set \( p_1 = \frac{4g_m V_{ref}}{k_3 C} \) and the equation becomes:

\[
y = (1 - (\frac{2g_m V_{ref}}{s \cdot k_3 C + 2g_m V_{ref}})^2)x + K'
\]

(4.24)

Recall in the signal-to-noise ratio equation we derived in (4.16), the SNR in dual-pole system is improved to:

\[
SNR = \frac{3k_3}{4V_{ref}H(\omega_{2nd})} = \frac{3k_3}{4V_{ref}} \frac{\omega_{2nd}C}{\omega_{2nd} + p_1} \approx 3\left(\frac{k_3 \omega_{2nd} C}{4V_{ref}g_m}\right)^2 = 3\left(\frac{\omega_{2nd}}{p_1}\right)^2
\]

(4.25)

If we assume the time constant of a second pole system \( \tau_{dual} = k_{dual} \frac{1}{p_1} \), where \( k_{dual} \) is the coefficient of time constant estimation from signal-pole to dual-pole system, then the new time-constant-to-SNR relationship can be written as:

\[
\tau_{dual} = \frac{k_{dual}}{\omega_{2nd}} \sqrt{\frac{SNR}{3}}
\]

(4.26)

The same analysis can be done for tri-pole systems. However, to validate the attenuation function from the addition of lowpass filters, it must be ensured that their cut-off frequency does not exceed the harmonic frequency, which means the settling time eventually is limited by the sinusoidal frequency whilst the harmonic fluctuation can be successfully restrained. In a tri-pole system, the combined lowpass transfer function \( H(s) \) becomes \( g_m s^2 C + b + 1 \), and the critical damping situation takes place when:

\[
\begin{align*}
  a &= \frac{k_3^2 C^2}{2g_m V_{ref}^2} \\
  b &= \frac{3k_3 C}{3g_m V_{ref}}
\end{align*}
\]

(4.27)

and the three equal roots are:

\[
p_{1,2,3} = \frac{1}{b} = \frac{3g_m V_{ref}}{k_3 C}
\]

(4.28)

Hence the SNR can also be defined by the following equation assuming the square term dominates:

\[
SNR = \frac{3k_3}{4V_{ref}H(\omega_{2nd})} \approx 3\left(\frac{\omega_{2nd}}{p_{1,2,3}}\right)^3
\]

(4.29)

and the TC-SNR relationship can be written into:

\[
\tau_{tri} = \frac{k_{tri}}{\omega_{2nd}} \sqrt{\frac{4}{3}SNR}
\]

(4.30)

The coefficient \( k_{tri} \) equals to 3.26, 2.31 and 1.825 for 63.2%, 90% and 99% of final values respectively. Simulation results of a tri-pole system are shown in figure 4.10. As can be
seen, a 50dB (0.048Np variation) accuracy is achieved without sacrificing the settling time. Applying an even higher order filter may have better SNR, but it will largely complicate the analysis and more importantly it will not decrease the time constant significantly. Hence, we are going to add a second-order lowpass filter after the rectifier to optimize our AGC system.

\[ x_{10}^3/g_{237}/g_{22}^4 \]

\[ 3.5 \]

\[ 4 \]

\[ 4.5 \]

\[ 5 \]

\[ 5.5 \]

\[ 6 \]

\[ 6.5 \]

\[ 7 \]

\[ 7.5 \]

\[ 8 \]

\[ \times 10^3 \]

\[ Np \]

\[ \text{Fluctuation} = 0.048 \]

\[ \text{Time spent} = 23\text{us} \]

\[ \text{Step change} = 4.43 \]

\[ \text{Transient} = 23\text{us} \]

\[ 1.36 \]

\[ 1.37 \]

\[ 1.38 \]

\[ 1.39 \]

\[ 1.4 \]

\[ x_{10}^3/g_{237}/g_{22}^4 \]

\[ 7.58 \]

\[ 7.59 \]

\[ 7.6 \]

\[ 7.61 \]

\[ 7.62 \]

\[ 7.63 \]

\[ 7.64 \]

\[ 1.4 \]

\[ 10^{-7} \]

\[ 1.36 \]

\[ 1.37 \]

\[ 1.38 \]

\[ 1.39 \]

\[ 1.4 \]

\[ \times 10^{-7} \]

\[ 3.5 \]

\[ 4 \]

\[ 4.5 \]

\[ 5 \]

\[ 5.5 \]

\[ 6 \]

\[ 6.5 \]

\[ 7 \]

\[ 7.5 \]

\[ 8 \]

\[ \text{Time(s)} \]

\[ \text{Natural Logrithmic Control Voltage (Np)} \]

\[ \text{Figure 4.10: Transient step response of a tri-pole system} \]

4.2 Practical consideration

In the previous section, we discussed the ideal calibration system which is challenged by several practical issues, and we will analyse them in the following subsections.

4.2.1 Offset upon Peak Detection

The peak detection function introduced before can always invert the negative signals and build the relationship between the DC strength after rectification and original sine wave amplitude. In reality, however, the input offset voltage will change the inverting point by certain amount and hence alters the waveform, shown in figure 4.11.
Mathematically the influence of offset can be derived as:

\[ |A_{cali} \cdot \sin(x) + K_{os}| = \sqrt{(A_{cali} \cdot \sin(x) + K_{os})^2} \]  

(4.31)

where \( A_{cali} \) is the calibrated gain which is the product of sensor sensitivity \( A_{sensor} \) and control gain \( A_{ctrl} \), and \( K_{os} \) is normalized value of input offset. When we focus on the terms in bracket we find the components as:

\[ A_{cali}^2 \cdot \sin^2(x) + 2A_{cali}K_{os}\sin(x) + K_{os}^2 = \frac{1}{2}A_{cali}^2 - \frac{1}{2}A_{cali}^2\cos(2x) + 2A_{cali}K_{os}\sin(x) + K_{os}^2 \]  

(4.32)

The target of the system is to keep \( A_{cali} \) constant which is doable for DC components and the offset term \( K_{os}^2 \) can be compensated by the external reference voltage.

Nevertheless, there is an AC signal depending on the offset voltage at the original frequency \( (2A_{cali}K_{os}\sin(x)) \). As described before, AC signals can also hamper the final accuracy of calibration, and here its effect is hoped to be at least less than the 2nd order harmonics:

\[ 2A_{cali}K_{os} \cdot H^2(\omega) < \frac{1}{2}A_{cali}^2 \cdot H^2(\omega_{2nd}) \]  

(4.33)

where the left part is the amplitude of original frequency compressed by the lowpass filter and the right part is that of the 2nd order spur. After certain derivation, we have the requirement as:

\[ K_{os} < \frac{1}{4} \frac{H^2(\omega_{2nd})}{H^2(\omega)} A_{cali} \]  

(4.34)

Assuming a third order lowpass filter is applied and then the middle term \( \frac{H^2(\omega_{2nd})}{H^2(\omega)} \) approximates to 0.17. As a result, the normalized offset value should be smaller than 4 percent of the amplitude of the calibrated sine wave:

\[ K_{os} < 4\% A_{cali} \]  

(4.35)
4.2.2 Input Noise & harmonics

Another critical issue in a real implementation is the distortion and noise of the input sine wave. The noise source can be the input-referred noise from the system itself, can be the front-end noise or even the chemical noise from solution.

As shown in figure 4.12, the variable gain amplifier outputs the product of the input sine wave \( S_1 \) and control signal (around 1), and the rectification can also be regarded as a multiplication of two same input. Hence it is obvious that the noise from the input sine wave at the end of the loop will appear on the control voltage. The direct consequence is that the resolution of calibration is limited by the SNR of the input sine wave.

By examining the rectification function, three major drawbacks could be found: firstly, the rectification halves the SNR of the signal from bandpass filter; secondly the non-idealities of the rectification generates certain spurs; and thirdly the self-reaction of the input signal generates plenty of inter-modulated components which can come back to the rectifier through the loop. To alleviate these issues, a chopper topology which periodically inverts the sine wave is preferred.

4.2.3 System parameters

Considering the limited bandwidth required for the ISFET front-end, the sine wave frequency is set to be 100kHz, which means the second order harmonic component is at 200kHz. The amplitude selection of the sine wave is a trade-off between input linear range and SNR to the calibration system, so an empirical value of 100mV\(_{pp}\) is chosen. We set the accuracy of the calibrated sensors sensitivity to be 1%, leading to a SNR of 50dB for abundance. A third order lowpass filter is applied due to its high efficiency and relatively low cost. As a
result, three identical poles to achieve -50dB harmonic attenuation are calculated to be:

\[
\left( \frac{f_c}{200k} \right)^3 < -50dB \Rightarrow f_c < 29k
\] (4.36)

Assume we set the cutoff frequency to be 25k, and recall equations 4.27 and 4.28 we have:

\[
\begin{cases}
p_c = \frac{1}{b} = \frac{3g_m V_{ref}}{k_3 C} \\
a = \frac{k_2 C^2}{27g_m V_{ref}} = \frac{1}{3p_c^2} = 13.65 \times 10^{-12} \\
b = \frac{k_3 C}{3g_m V_{ref}} = \frac{1}{p_c} = 6.37 \times 10^{-6}
\end{cases}
\] (4.37)

Because the proposed system performs a closed feedback loop with an external reference voltage, certain process variation on absolute values can be feasibly adjusted by changing this external reference and will not affect the system.

4.3 System Implementation

In the system realization in CMOS, we integrate the exponential function into the Variable Gain Amplifier A(x), followed by a sine wave extraction stage, which impose a pre-amplification of a factor of approximately 8. The rectification function is realized through a chopper in current mode. The clock used in the chopper is also generated from the sine wave to ensure minimum phase difference. Because of the feasibility of the arithmetic operation of currents, the reference signal is converted to current mode as well. Finally, since the sine wave signal may change in a large range after the amplification, we exert the lowpass filtering in current mode again to avoid large voltage swing.

![Diagram of implementation of AGC system](image)
Figure 4.13 shows the implementation of our AGC system, whose details will be introduced in the following sections.

4.3.1 Variable Gain Amplifier

The most challenging part of a CMOS implementation of our desired VGA is the exponential function due to the inherent square law characteristics in CMOS technology. The exponential characteristic in weak inversion region is not suitable because of its low bandwidth. Fortunately, unlike the VGA used in communication systems, where large tunable range (70dB) and high bandwidth are required [6, 7], amplifiers applied in our system are mainly responsible for the sensitivity calibration which has a more relaxed tuning range requirement.

As a result, we could use the pseudo-exponential function expressed as $e^x \approx (1 + x)/(1 - x)$ [8-12]. Besides that, the absolute readout values from front-end ISFET sensors may vary according to the bias current, pH values and temperature, so the VGA should have a good CMRR and large input linear range.

Figure 4.14 shows the design methodology of the VGA, where two identical transconductors $OTA_{1,2}$ operating as the most conventional active VGA [13] are designed to have large input range and the fully differential transconductor $OTA_3$ controls the total gain.

![Variable Gain Amplifier Diagram](image)

Figure 4.14: Block level structure of the variable gain amplifier

Since the amplification of the VGA is defined as $g_{m1}/g_{m2}$, which can be derived in terms of bias current:

$$Gain = \frac{I_{b1}(\frac{g_{m1}}{I_{d1}})}{I_{b2}(\frac{g_{m2}}{I_{d2}})}$$  

(4.38)

Assuming the desired gain is around unit gain, the DC condition of the two transcon-
ductors $OTA_{1,2}$ should be the same, leading to the same value of $(\frac{g_m}{I_d})_1$ and $(\frac{g_m}{I_d})_2$. So the gain approximates to the ratio of the two bias currents $I_{b1}/I_{b2}$. Again, if we assume the control voltage $V_{ctrl}$ locates besides zero, we can obtain the relationship between the control voltage and gain:

$$Gain = \frac{I_{b1}}{I_{b2}} = \frac{I_{dc} + \Delta I}{I_{dc} - \Delta I} \approx \exp\left(\frac{\Delta I}{I_{dc}}\right) = \exp\left(\frac{g_{m3}}{I_{b3}} \Delta V_{ctrl}\right)$$  \hspace{1cm} (4.39)

Compare equation 4.8 and 4.39, we found the coefficient to be:

$$k_3 = \frac{I_{b3}}{g_{m3}}$$  \hspace{1cm} (4.40)

**OTA 1&2**

Because the VGA is the first stage of the calibration system, there are certain requirements for the OTA comprising the VGA: they have to be low noise and have a large input range.

To increase the input range, we applied the Bump Linearisation technique on the conventional folded OTA [14]. When the W/L ratio of $M_{n3}$ and $M_{n4}$ is twice of that of $M_{n1}$ and $M_{n2}$, the third order Taylor series term of this OTA can be cancelled which largely improves the linearity. We further split $M_{n3}$ and $M_{n4}$ into two identical transistor to ensure a symmetrical bump for both positive and negative inputs. The specification of the OTA is summarised in the table 4.1.

<table>
<thead>
<tr>
<th>$M_{p1,2}$</th>
<th>$M_{n1,2}$</th>
<th>Bias current</th>
<th>$g_m$</th>
<th>Bandwidth (1k-1M)</th>
<th>Input noise</th>
<th>THD(1%) @100k</th>
</tr>
</thead>
<tbody>
<tr>
<td>20μm/12μm</td>
<td>4μm/20μm</td>
<td>20μm</td>
<td>31.5μm</td>
<td>2.4MHz</td>
<td>28.6μV</td>
<td>476.4mVpp</td>
</tr>
</tbody>
</table>

**OTA 3**

According to the analysis we have done before, the control sensitivity is determined by the third OTA as $k_3 = \frac{I_{b3}}{g_{m3}}$. Recall the equation to calculate the cut-off frequency $p_c = \frac{3g_{mV}_{ref}}{e_kC}$, a larger coefficient $k_3$, or in other words smaller $\frac{g_{m3}}{I_{b3}}$, allows us to use a smaller load capacitor which is desirable in CMOS implementation.

On the other hand, although the noise introduced by the sine wave is independent on
the choice of $k_3$, larger $k_3$ makes the VGA less sensitive to the random thermal noise. As a result, the value of $k_3$ is tuned to be 0.61, indicating the value of $\frac{k_{ma}}{g_{m1}}$ to be 1.64.

4.3.2 Sine wave extraction & Pre-amplification

Because the sine wave is located at a relative high frequency, 100k Hz, compared to the chemical signals, it is feasible to use a highpass filter to extract the sine wave. Considering the realistic offset voltage and its requirement comparing to the input sine wave ($K_{os} < 4\%A_{cal}$), a sine wave of at least 400$mV_{pp}$ is desired. The input sine wave however is also limited to the input range, and only 100$mV_{pp}$ is affordable. Hence certain amplification implemented by a low-noise amplifier is required, shown in figure 4.16.

Thanks to the large frequency gap between electrical and chemical signals, a single-stage RC filters should be enough. A second highpass filter is used to remove the input offset of the amplification OpAmp. The amplification ratio is decide by the input and feedback resistors which is set to 8. Because we realise the tunability in the VGA, gain or cutoff frequency deviations at this stage are not of interest.

The amplified signal should have a 800$mV$ swing range, which imposes a strict requirement on the input linear range in the next block. A feasible approach that converts the
4.3.3 Amplitude Detection

The amplitude is detected through the method we mentioned before: convert the voltage signal into current domain and then chop the current to invert negative signals. The block-level structure is shown in figure 4.17. Again, the lowpass filter operates in the current domain as well.

Figure 4.16: Schematic of Pre-amplifier with two highpass filter

Figure 4.17: Block structure of the amplitude detector
V-I OTA

The value of transconductance of the OTA ($g_m$) is shown in equation 4.28, and we wish to keep the $g_m$ value small to avoid using a larger load capacitor. At the same time, too small current hampers the accuracy of the rectification realized by chopping. Another fundamental challenge is the input range where we expect a 800mV swing. Because of all these reasons, we used a resistive voltage to current converter as the OTA we needed [15].

As shown in figure 4.18, transistors $M_{n0a,b}$ and $M_{p0a,b}$ provide the same bias current to two branches, and they should be as matched as possible to decrease the input offset. Input differential pair $M_{p1a,b}$ are simply voltage shifters and impose the differential input voltage upon a resistor $R$. The current flowing through the resistor leads to a different current to flow through transistors $M_{n1}$ and $M_{n2}$ proportional to the input voltage. Transistors $M_{n1,m}$ and $M_{n2,m}$ are the current mirrors of $M_{n1}$ and $M_{n2}$ respectively with a size ratio of 1/5. Hence, the transconductance of OTA can be derived as $\frac{2}{5R}$. The specification of this V-I OTA is summarized in table 4.2:

Phase Generator

Due to the uncertainty of the phase delay of the sine wave through chemical solution, the chopper clock is generated from the amplified sine wave mentioned in the last section. The schematic of this block is shown in figure 4.19.
Table 4.2: Specification of the transconductor

<table>
<thead>
<tr>
<th>$g_m$</th>
<th>$1\mu A/V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{p0}$</td>
<td>$15\mu m/30\mu m$</td>
</tr>
<tr>
<td>$M_{p1}$</td>
<td>$100\mu m/2\mu m$</td>
</tr>
<tr>
<td>$M_{n0}$</td>
<td>$5\mu m/80\mu m$</td>
</tr>
<tr>
<td>$M_{n1,2}$</td>
<td>$10\mu m/40\mu m$</td>
</tr>
<tr>
<td>$M_{n1m,2m}$</td>
<td>$2\mu m/40\mu m$</td>
</tr>
<tr>
<td>$R$</td>
<td>$400k\Omega$</td>
</tr>
<tr>
<td>$I_{b,tot}$</td>
<td>$9.5\mu A$</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>$1.08MHz$</td>
</tr>
<tr>
<td>THD(1%)</td>
<td>$922mV_{pp}$</td>
</tr>
</tbody>
</table>

Figure 4.19: Schematic of phase generator

A two non-overlapping clock is generated through the inverter chain and a certain phase shift is applied through a RC network. According to the parameters shown in the figure, the phase shift is:

$$\phi(100k) = 90^\circ - \tan^{-1}\left(\frac{2\pi 100k}{f_c}\right) = 90^\circ - \tan^{-1}\left(628k \times 8.2\mu\right) = 11^\circ \quad (4.41)$$

What we care about more is the deviation of phase shift according to the component mismatch, and hence we derive the derivative of phase as:

$$\frac{\delta \phi(\omega)}{\delta \omega_0} = \frac{1}{1 + \left(\frac{\omega}{\omega_0}\right)^2} = \frac{\omega}{\omega_0^2 + \omega^2} \quad (4.42)$$

If we normalize the cutoff frequency to unit frequency, we can have the following equation:

$$\rightarrow \frac{\delta \phi(\Omega)}{\delta \Omega_0} \mid_{\Omega=5.15} = 0.187 \text{ rad} \quad (4.43)$$

As a result, a 10% deviation at pole frequency will only cause $0.187 \cdot 0.1 \cdot 360^\circ = 1.07^\circ$ phase shift, which is more than acceptable.
Current Mode Lowpass Filter

The specification of this lowpass filter, orders and cutoff frequency have been decided from the top system design since it serves the harmonic suppression. As we discussed above, the transfer function is chosen to be:

\[
H(s) = \frac{1}{13.65 \cdot 10^{-12} \cdot s^{2} + 6.37 \cdot 10^{-6} \cdot s + 1}
\]  

(4.44)

In addition to the decent high frequency performance, current mode approach can also largely simplify the circuits thanks to its easy implementation of multiple outputs such as dual-output OTAs [16-18]. Because of the feasibility of multiple outputs, a design methodology of applying multiple integrator loops are widely applied [19, 20]. One of the implementation of a second order current mode lowpass filter is shown in figure 4.20. This topology was chosen due to its simple structure which allows multiple outputs for higher order integration loop.

![Figure 4.20: Basic current mode lowpass filter by dual-output OTA-C structure](image)

According to the figure, we could have equations as:

\[
\begin{align*}
V(I_{in}) \cdot sC_{1} + I_{1} + I_{2} = I_{in} \\
V(I_{in}) \cdot g_{m1} = I_{1} = V_{t} \cdot sC_{2} \\
V_{t} \cdot g_{m2} = I_{2} = I_{out}
\end{align*}
\]

(4.45)
\[ I_{\text{out}} = \frac{g_{m1} g_{m2}}{s^2 C_1 C_2 + s C_2 g_{m1} + g_{m1} g_{m2}} I_{\text{in}} = \frac{1}{s^2 \frac{C_1 C_2}{g_{m1} g_{m2}} + s \frac{C_2}{g_{m2}} + 1} I_{\text{in}} \]  

(4.46)

Clearly, the terms \( \frac{C_1 C_2}{g_{m1} g_{m2}} \) and \( \frac{C_2}{g_{m2}} \) stand for coefficients \( a \) and \( b \) in equation 4.27, and should equal to \( 13.65 \cdot 10^{-12} \) and \( 6.37 \cdot 10^{-6} \) respectively. To simplify the design, we set the transconductance of OTAs to be the same value as the Amplitude Detection block (1\( \mu \)), hence \( C_1 = 2.13pF \) and \( C_2 = 6.4pF \).

### 4.4 Full AGC measured results

Since the system is designed for a large sensor array, the layout of it is kept to a long strip as shown in figure 4.21.

![Figure 4.21: The layout of proposed system including output buffers](image)

In the following subsection, different specifications are tested.

#### 4.4.1 Noise performance

The noise test is conducted under the scenario that a constant sine wave (100kHz \( \pm 100mV_{pp} \)) is driven at the input. From figure 4.22 we can see the peak-to-peak value of the output of the control voltage is 7mV which indicates a 1.15% gain deviation given by \( \frac{g_{m3}}{I_{b3}} \).

However, a closer investigation shows that the peak-to-peak fluctuation does not only come from second order harmonic (200k component). By analysing the frequency spectrum of this control signal shown in figure 4.23, it is clear that the low frequency and original components also contribute considerable noise due to the practical issues we discussed before, such as noisy sine wave input and input-referred offset. In terms of the second harmonic spurs, it has been compressed well. The tested data (-63.55dB) indicates there is only 1.33mV\(_{pp}\) 200k component.
4.4.2 Settling time

Figure 4.24 shows the tested results for step input changes with 20% modulation depth. The solid black sine wave is the calibrated output signal responding to an amplitude modulated input which is marked by dotted black line. The red line is the differential control voltage. Due to the high noise level on the control voltage, we average the periodic waves for more distinct responding tracks. Though we observed some overshoot of the control voltage, we could still tune the amplification within roughly 70μs.
4.4.3 Calibration accuracy

The calibration accuracy of the proposed system to varying systematic errors in addition to the random noise already measured is tested and evaluated. Sine waves with different amplitudes varying from 70mV to 130mV were driven to the system. The amplitudes of the measured output sine waves have always successfully reached an amplitude of 97.6mV.
Automatic Gain Control System for ISFET array compensation

The variation of the corresponding control voltage for these input sine waves is shown in figure 4.25. This confirms the AGC works as expected and tolerates systematic errors.

4.5 Conclusion

In this chapter, a novel automatic sensitivity calibration system has been proposed. It utilises the high frequency sine wave signal superimposed on the low spectrum chemical signal to estimate the sensitivity of each single sensor pixel. By rectifying the sine wave and filtering out the AC part, the remaining DC value can be obtained which should be proportional to its original amplitude. The calibration is executed through a negative feedback which has exponential response according to input changes. This linear-in-dB relationship could guarantee a constant response time for inputs with different amplitudes. Although the settling time still costs certain time before sensing, it is necessary to compensate for the gain mismatch in our previously proposed ISFET interface.

What makes this calibration system different to others is its specific requirements: limited input range but higher accuracy. Thanks to the small input range, the exponential relationship can be implemented through an approximation \( e^x = (1 + x)/(1 - x) \). The high accuracy requirement, on the other hand, triggers much more practical consideration. The fundamental one comes from the systematic limit due to the second order harmonics after rectification which exerts a trade-off between speed and SNR. To address this dilemma with minimal cost, additional filters are added. A third-order lowpass filter in total is chosen considering the performance and complexity. Non-ideal issues in circuits design such as input-referred noise and offset have also been taken into account.

The tested result shows that the system can be very accurate after certain averaging, but low frequency noise caused by imperfection input sine wave still largely deteriorates the final SNR. The settling time for 99% accuracy is found to be around 70.5\( \mu \)s which is similar to our simulation result. Finally, the summary of the system has been listed in table 4.3.
Table 4.3: Designed specification of the whole AGC system

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>3.3V</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>4.8mW</td>
</tr>
<tr>
<td>Area</td>
<td>0.37mm²</td>
</tr>
<tr>
<td>( k_3(I_{b3}/g_{m3}) ) in Variable Gain Amplifier</td>
<td>0.61</td>
</tr>
<tr>
<td>( g_m ) in Amplitude Detection block</td>
<td>1µA/V</td>
</tr>
<tr>
<td>( C_{load} )</td>
<td>8.09pF</td>
</tr>
<tr>
<td>( C_1 ) in Lowpass filter</td>
<td>2.13pF</td>
</tr>
<tr>
<td>( C_2 ) in Lowpass filter</td>
<td>6.4pF</td>
</tr>
<tr>
<td>Reference voltage ( V_{ref} )</td>
<td>252mV</td>
</tr>
<tr>
<td>Settling Time (99%)</td>
<td>( \approx 70.5\mu s )</td>
</tr>
<tr>
<td>Signal-to-Noise Ratio</td>
<td>36.5dB</td>
</tr>
</tbody>
</table>
Bibliography


Chapter 5

An ISFET chemical sensing array with automatic gain calibration

This chapter presents a full system composed of an ISFET sensing array, an automatic gain calibration system, AD/DAs, SPI (Serial Peripheral Interface) connections and other digital logic to achieve an accurate chemical sensing platform. The sensing array comprises $32 \times 32$ pixels using the single-stage ISFET front-end introduced in Chapter 3. The automatic gain calibration system is also described in Chapter 4. Other auxiliary circuits are designed to bridge these two main blocks.

It is envisaged that the designed system achieves the following novelty:

- ISFET offset voltage distribution due to trapped charge is minimised
- Gains of the sensor pixels over the whole array are consistently guaranteed through the AGC
- Low power implementation for each pixels to allow power efficient scaling for large array
- A scalable architecture to allow fabrication of large arrays

This array can be integrated into a DNA sequencing platform and used in the following way: the DNA beads are scattered on the surface of the sensor array, and each bead going in one well on top of a number of sensors. Then an initial calibration process can be executed to cancel the drift and trapped charge. Afterwards, the reagents are flowed through the chip
to normalise the reaction of each nucleotide incorporation. Since the calibration process is real-time, we just need to reset the DC point every time before a new nucleotide flow.

This chapter proceeds with top level architecture where the functionality of each block is given. Then attention is given to the implementation of each block including the circuits specifications. The complete fabricated system is then presented. Finally the simulation and measured results are given and discussed.

5.1 System architecture

The top level architecture is shown in figure 5.1. The sensors in the array are addressed through multiplexing. Each sensor will be calibrated in real time by the Automatic Gain Calibration system proposed before. A digital CDS (Correlated Double Sampling) scheme is applied to deal with the offset of each pixel. All the digital signals are controlled by a central control block to realize synthesised operation. SPI blocks send the digitalised signal off chip.

![System Architecture Diagram]

Figure 5.1: System Architecture
5.1.1 Pixel array

The pixel array works in a mechanism similar to that of a RAM. Pixels are firstly selected row-by-row by an address decoder, whose outputs are then picked up by a column header, shown in figure 5.2. The current mirrors in all pixels are biased globally, but the reset signal is buffered and driven to single rows only to guarantee a fast slew rate, which is critical to minimise the charge injection effects.

![Diagram of Pixel Array Readout Architecture](image)

Figure 5.2: Pixel array readout architecture

5.1.2 Digital CDS

The electrical offsets cancellation is achieved by designing a double sampling process in digital domain. Due to the requirement of medium & long-term measurements (above 4 seconds), using analogue memory is not a reliable approach due to leakage, therefore a digital CDS is adopted. In addition to that, the elimination of the need of capacitors also makes digital CDS attractive in some applications, such as imaging arrays [1, 2].

We could consider digitalising the chemical signal and subtracting the offset in digital domain, but usually the ranges of the electrical offset and the chemical signal differ largely (e.g. 200mV for offset ADC whilst 2V for sensing ADC), which requires an internal scaling
An ISFET chemical sensing array with automatic gain calibration

accordingly, and the reference voltage has to be sufficiently accurate. Therefore we convert the digital offset signals back to analogue ones and execute the subtraction in analogue domain.

The offset signal is sampled after a lowpass filter, used to remove the high frequency sine wave from the AGC output, which stores all the non-ideal information such as offsets in sensor pixel, AGC, lowpass filter and the different calibration factor:

\[
V_{store} = (V_{os,pixel} + V_{osi,AGC}) \cdot A_{cali} + V_{oso,AGC} + V_{os,LP}
\]

(5.1)

where \(V_{osi,AGC}\) and \(V_{oso,AGC}\) stand for the input-referred and output-referred offset, of which the former varies according to the amplification while the latter is independent of it.

The digital offset data are sent out and stored in the off-chip memory due to the scarcity of chip resources. Every time before a pixel is selected its offset data are fetched back from that memory to ensure correct chemical signal readout.

5.1.3 AGC & Lowpass filter

The automatic gain calibration system is the same as the one described in Chapter 4. Recall the system implementation in figure 5.3, the output comes from the variable gain amplifier. Hence the output signal comprises both low frequency chemical signal and high frequency sine wave signal, which requires a lowpass filter to select the chemical signal only.

![Figure 5.3: Implementation of the automatic gain calibration system](image-url)
5.1.4 Digital control unit & SPI

The central digital control unit synthesises the operation of all other blocks, including the multiplexing of sensor array, AD/DA and the SPI block. Since each multiplexing action needs some time for settling and calibration, it is better to utilise this period for the SAR ADC process, and figure 5.4 demonstrates a scenario of offset sampling. Whenever an AD conversion starts, the current analogue offset signal is sampled and held, hence it is acceptable to switch to the next pixel by changing the pixel address. As long as an acknowledgement signal is received from the ADC, the offset has been digitalised and ready to be sent off chip. Since the current pixel has been selected for certain period (8 clock cycles here), another AC conversion can start immediately. The same concept is applied for other scenarios such as reading the chemical signals.

![Timing graph of digital control logic](image)

Figure 5.4: Timing graph of digital control logic

Before the offset or signal data is sent out through the SPI, it is attached with its own address number. As a result, each sent word contains both the address and data, which facilitates the data processing in work stations.

5.1.5 Instrumentation Amplifier

The offset signal is retrieved and converted back to analogue domain. To subtract it from the pixel readout, a low offset instrumentation amplifier is implemented, whose output presents the final analogue output and can also be further digitalized. To amplify the input changes, an amplification of 3 is set in the instrumentation amplifier.
5.2 System Implementation

5.2.1 ISFET pixel schematic

As discussed in chapter 3, the single-stage ISFET front-end achieves a better balance between power consumption, accuracy and functionality. As a result, this structure is chosen to build the array. A $32 \times 32$ array is implemented where routing capacitance becomes a problem causing a limitation in bandwidth. Since certain bandwidth is required to transfer high frequency sine wave signals, a local buffer with power-down ability is added, as shown in figure 5.5.

![Schematic of pixel in the array](image)

Figure 5.5: Schematic of pixel in the array

Another auxiliary circuit added is the local reference set-up branch. As was described previously, the key to achieve an extremely low leakage switch is to keep the difference between the stored and reference voltages minimal. Nevertheless, the dimension of the whole array is large enough to take into account process variations, hence a local reference is preferred. As shown in figure 5.5, the reference voltage for switch $V_r$ is locally generated.

Apart from these two features mentioned, the specifications of the ISFET sensor are the same with those described in chapter 3. The layout of each pixel is shown in figure 5.6, and the decoupling capacitor is optional, but has been added to improve PSRR.
5.2.2 Lowpass Filter

Since the output signal from the AGC shown in figure 5.3 consists of both low frequency chemical and high frequency sine wave signals, it is necessary to filter out the sine wave component meantime keep the system responding fast. An intuitive method is applied here, six 1st-order lowpass filters with 50kHz cut-off frequency are connected serially. Figure 5.7 shows the structure of the filter, where each gm stage equals to $290nA/V$ and the load capacitance is around $900fF$. By doing so, an attenuation of 125 at 100kHz is realised.
5.2.3 Instrumentation Amplifier

A conventional Instrumentation amplifier is implemented here. Given that the linear range of the AGC output is around 800mV, it is acceptable to add certain amplification in the stage to magnify the chemical changes. According to the implementation shown in figure 5.8, the gain of it is:

\[ G = \frac{R_4}{R_3} \left( 1 + \frac{2R_1}{R_2} \right) \]  

(5.2)

By setting all the resistor values to be 200kΩ, the amplification is calculated to be 3.

5.3 Fabricated system

The whole system was designed and fabricated in AMS 0.35µm CMOS technology. The fabricated chip is shown in figure 5.9 followed by annotations of its floor plan. Due to the requirements for glob-top which isolates the metal pads and wire-bonding from the solution, the ISFET array is located in the centre of the chip and at least 200µm gap is left from each edge. The analogue and digital pads are separated at opposite sides of the chip and two on-chip decoupling capacitors are added to improve the noise performance.

The SPI is integrated in the digital control block at the corner of the chip.
5.4 Test set-up

The test set-up is shown in figure 5.10. The bare CMOS dies were wire-bonded and epoxy encapsulated on a small PCB base which is then stacked on top of another PCB board. The
large PCB test board assembles various components including voltage regulator, variable resistors and analogue buffers to provide clean power supply and reference voltages to the chip under test. There are connection headers on each side of the big PCB, one for digital signal extension and the other for analogue monitoring with several SMB ports.

![Diagram](image)

(a) Demonstration of the test set-up

(b) Photo of the test set-up

Figure 5.10: Test set-up for the system

A micro-fluidic flow cell is champed on top of the small PCB board forming a sealed chamber except for the inlet and outlet. The whole chip is immersed in solution whose flow is controlled by a syringe connected with the inlet of the chamber. An Ag/AgCl reference electrode is placed in the centre of the chamber to bias the ISFET devices. Because the chamber is fully sealed, new solution coming from the inlet will expel any previous solution rather than reacting with it and the solution volume used is large enough to ensure this. As a result, the pH concentration the chip senses is the same as that kept in the syringe
when the chamber is flushed. Tricine 1M KCl pH buffers have been used for the subsequent experimental tests.

The off-chip memory is implemented in a DE0 FPGA board, which is also responsible for sending the data to a computer to demonstrate the output values in real-time. Due to the clock synchronisation requirement, the clock used in the chip is generated from the FPGA. The connection between FPGA and computer is based on the UART protocol.

5.5 Simulated Results

Post-layout Monte-Carlo simulation results are given in figure 5.11. To demonstrate the operation of the system, an ISFET array consisting $4 \times 4$ pixels was simulated instead due to the length of time required for a full system simulation. The passivation capacitors of these pixels are randomly allocated between $77\,\text{fF}$ and $107\,\text{fF}$ to simulate the mismatch while the feedback capacitor is $72\,\text{fF}$. The clock frequency set in the simulation is $100\,\text{kHz}$.

As can be seen in the figure, the sensing process starts with an offset cancellation step, where each pixel is sampled with the gain calibration system. After the offset sampling process, the equivalent chemical input voltage is changed by $100\,\text{mV}$, which takes place at $2.5\,\text{ms}$ in the graph. Afterwards, the offset data is fetched back every time before reading the chemical signal. During the multiplexing of pixels, the control voltage can always settle quickly to calibrate the gain deviation. The bottom figure shows the final outputs, whose distribution values are shown in figure 5.12. The simulation results shows a $2.4\%$ given deviation, whose original number before calibration is $39\%$. Therefore we confirm the functionality of the system.

5.6 Measured results (Fabricated chip)

5.6.1 Pixel DC offsets

The first function of the total system is the DC offset cancellation in digital domain. By fetching out offset data from the off-chip memory, a normal distribution of reset voltages is represented in figure 5.13.

A standard deviation of $14.3\,\text{mV}$ of the DC output voltage is calculated which indicates the cancellation of trapped charge was successful. The mean value is less important because
it is tunable via the source voltage of all the pixels. From the following test results it can be seen that all the pixel readouts start from the same voltage level after this offset cancellation.

Table 5.1: Standard deviation of offset voltages over chips

<table>
<thead>
<tr>
<th>Chip 1</th>
<th>Chip 2</th>
<th>Chip 3</th>
<th>Chip 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std</td>
<td>13.9mV</td>
<td>14.2mV</td>
<td>14.0mV</td>
</tr>
</tbody>
</table>
Figure 5.12: The final output values for different pixels

Figure 5.13: The distribution of the DC voltages over 4 chips

step shown in figure 5.14.
5.6.2 Drift due to switch leakage

As discussed before, the main concern of the two-phase sensing mechanism is the leakage due to the switch. The sensor array however will also suffer from different leakages in different pixels. Figure 5.14 shows how the leakage is tested, the output voltage is recorded with no input changes. The effect of the leakage current over the whole array has also been amplified by a factor of 3 in the instrumentation amplifier stage.

By investigating the leakage situation in different dies, we found that its pattern still follows a normal distribution and mean values approximate to zero. The drift rate is shown in figure 5.15

Figure 5.15: Distribution of drift rate due to switch leakage of the whole chip
5.6.3 Chemical Test

![Voltage output (V) graphs for different times and pH values.]

Figure 5.16: pH image of the whole sensor array from pH 9 to pH 7

Since the sensor architecture is the same as the one tested previously, characterisation of the individual pixel can be found in Chapter 3.

The chemical sensitivity test is the same as described in chapter 3, pH 9 buffer is flushed...
into the chamber to replace the existing pH 7 buffers imitating a pH change. Figure 5.16 shows a 3-D snapshot of the output of the array. Progressively in time as the pH 7 solution is replaced by pH 9, what we see is a clear change in output response as expected.

The statistical data in this pH change process is shown in figure 5.17. Considering the in-pixel amplification of 1.26 and the amplification in instrumentation amplifier (3), the intrinsic sensitivity of the sensor is approximately 39.6mV/pH. This number is different with the values we tested before, we argue that the devices tested come from different fabrications runs. Inaccuracy in pH buffer concentration could also contribute to this. But this value meets our expected sensitivity based on characterisation data from this process.

![Figure 5.17: Voltage outputs of the array before and after the flowing of the solution](image)

**ISFET surface sensitivity experiment**

To test the sensitivity of surface material of the fabricated ISFET array when it is modified, a drop of Sodium Hydroxide is left on the array for 4 hours to corrode the Silicon Nitride passivation layer. By doing so, a distinctive image inconsistency is represented in figure 5.18 when pH 9 buffer is flowed to replace the pH 7 buffer.
The gain of sensors over the whole array is tested by changing the reference electrode voltage. When the reference voltage was increased by 200mV, we found a consistent response at the outputs of the sensor array, shown in figure 5.19 where the output changed by around 700mV.

The effect of the AGC system is tested in the same way, where the reference voltage was increased by 200mV. The original and calibrated output signals are compared over 4 chips, whose statistics are shown in figure 5.20. As can be seen, because the original gain distribution is much better than expected in this fabrication run, we only see a slight improvement as shown by the results in table 5.2. This indicates our results will give a lower spread in output but never worse, even if the array is worse. An important point to note during the test process is the leakage from the switch makes it challenging to get higher resolution results. For example, if a test step takes 10 seconds, then the voltage changes due to the leakage can be up to ±20mV according to our previous tests, while the total reference change is only 200mV, indicating up to 10% variation. The data in figure 5.20 is calculated based on the subtraction of leakage drifts.
Figure 5.19: Amplification test shows a good consistence over the whole chip.

Figure 5.20: The distribution of the gain over 4 chips.
Table 5.2: Standard deviation of the gain of pixels over chips

<table>
<thead>
<tr>
<th>Chip</th>
<th>Original</th>
<th>Calibrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>chip 1</td>
<td>17m</td>
<td>9.8m</td>
</tr>
<tr>
<td>chip 2</td>
<td>10.9m</td>
<td>9.1m</td>
</tr>
<tr>
<td>chip 3</td>
<td>11.3m</td>
<td>10.2m</td>
</tr>
<tr>
<td>chip 4</td>
<td>12.5m</td>
<td>10.8m</td>
</tr>
</tbody>
</table>

5.7 Conclusion

In this chapter, a system combining the ISFET sensing array and the automatic gain calibration system is presented with their achieved specifications listed in Table 5.3. Digital correlated double sampling is applied to cancel the DC mismatch of pixels. Due to the on-chip resource limitation, digitalised offset data is stored in off-chip memory whose writing & reading operation is executed via an SPI protocol.

Table 5.3: Specifications of the full system

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>AMS 0.35μm 2P4M CMOS</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>3.3V</td>
</tr>
<tr>
<td>Die dimension</td>
<td>2.7mm × 2.35mm</td>
</tr>
<tr>
<td>Array Size</td>
<td>32 × 32</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>2.66mW (Array) (1.98μW/pixel)</td>
</tr>
<tr>
<td></td>
<td>0.52mW (Digital logic @ 1MHz clock)</td>
</tr>
<tr>
<td></td>
<td>4.65mW (AGC)</td>
</tr>
<tr>
<td></td>
<td>2.29mW (Others)</td>
</tr>
<tr>
<td></td>
<td>10.12mW (Total)</td>
</tr>
<tr>
<td>Sensing frame</td>
<td>9.3 frames/sec (105μs/pixel)</td>
</tr>
<tr>
<td>Trapped charge cancellation</td>
<td>$\sigma = 14.3mV$</td>
</tr>
<tr>
<td>Gain Calibration</td>
<td>$\sigma = 9.98mV/V$</td>
</tr>
<tr>
<td>Intrinsic sensitivity</td>
<td>39.6mV/pH</td>
</tr>
</tbody>
</table>

Test results show that the offset problem can be properly solved by the reset phase before sensing and the remaining electrical offset has only an average of 14.3mV standard deviation. The digital CDS could guarantee an adjacent DC output voltage for all the sensors, whose resolution is determined by the resolutions of both CDS and sensing ADCs. The leakage from the switch could be an issue but subtraction of it can to some extent alleviate the situation.

The advantage of the calibration system was not well emphasised. The in-pixel amplifi-
cations of all the pixels in the array are surprisingly good and the AGC system only achieves marginal improvements. The system might therefore be more beneficial for sensing arrays comprising 3-T ISFET structure, which suffers from larger gain deviations.

We have demonstrated that the complete 32x32 array has good performance for measuring pH as demonstrated by the specification achieved in table 5.3. In terms of further increasing the number of pixels, the main bottleneck remains in the pixel sizes whereas the AGC and other auxiliary circuits cost little area. To speed up the sensing frame, one possible solution would be executing the calibration process periodically rather than every time before sensing. That would leave the speed limit to the readout speed instead of calibration settling time.

With respect to the target application in DNA sequencing, the achieved performance is sufficient for hydrogen ion detection and offset variation has been managed to a good level considering the potential variation in floating gate voltage that would be present without compensation. This completes our work on the front end.

We now proceed in the next chapter to describe novel back end processing mechanisms to interface the sensors and do real-time DNA fragment comparison for genome assembly.
Bibliography


Chapter 6

Real-Time Sequence Comparison for DNA Assembly

6.1 Introduction

The length of DNA fragments that can be sequenced by current technologies commonly ranges from 200 to 700 bases, which is orders of magnitude shorter than the original genome. It is also common that the DNA short reads that are sequenced have no previous libraries with which to compare to, thus the restoration of the original DNA can only occur through recombination of the fragments. One solution for this is oversampling, which is to sequence the DNA with coverage of over 20-fold. This method is termed as the de novo approach. Based on the detected DNA data, assembly algorithms are responsible for the reconstruction of the fragments into the original genome.

The increase in detected DNA data, however, exerts an even bigger challenge to the assembly process. Finding the overlaps of fragments, which is the core requirement of assembly, has been classed as mathematical NP-hard problem and takes time to process [1]. In this chapter, we present a novel FPGA implementation for real-time DNA fragment overlap detection that can speed up the assembly process through a highly paralleled approach. It is envisaged that this system can be combined with the detection array introduced in chapter 5 to realise a fully automatic DNA detection and assembly system introduced in the first chapter.
6.2 Existing assembly algorithms

We now proceed to give an overview of an existing assembly algorithm. The majority of existing assembly algorithms are designed for PCs and are based on graphs [2-4]. They are classified into three categories: 1) the overlap-layout-consensus methods based on an overlap graph, 2) the de Bruijn Graph method and 3) the greedy graph algorithm which is a combination of the previous two. Due to the long time requirement for processing whole fragments and the corresponding required RAM space, a similar concept of using k-mer\(^1\) is applied in almost all the assemblers. To understand the adoption of these in a hardware system, we give a brief overview of these methods in the following section.

6.2.1 Greedy

The Greedy method and its variants, such as TIGR Assembler [5] and CAP3 [6], have been widely used even in the 1990’s when the Sanger method was the mainstream. Its relatively simple operation is what makes it so successful: individual fragments are joined together into contigs\(^2\) iteratively according to the overlapping neighbours. The term “best-overlapped” between two reads is defined as the instance when the suffix of one read matches the prefix of an other more than any others, according to program-specific criteria. Hence the contig will grow in a greedy way until no more fragments can be added. This approach optimises the local solution however is prone to missing the global answer, shown in figure 6.1. The figure also shows the algorithm producing two incorrect contigs by seeking the best overlaps.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.1.png}
\caption{Assembly produced by the greedy approach}
\end{figure}

\(^1\)A k-mer is a substring of the original sequence of length k

\(^2\)A contig is a set of overlapping DNA segments that together represent a consensus region of DNA.
To be more suited for the Next-Generation Sequencing platforms, this approach has been improved and other greedy strategies have been reported. For example, SSAKE [7] uses a lookup table of reads whose prefixes have been indexed, and overlaps are more carefully chosen for multiple extension cases; and based on the SSAKE method, SHARCGRS [8] adds a pre- & post-processing, where the former filters errors and latter merges contigs.

6.2.2 Overlap-layout-consensus

The OLC approach was the most successful assembly strategy for practical application as it can carry out the global analysis of all the fragments and had contributed a majority of large genomes assembled via platforms such as Celera [9], Arachne [10] or Newbler [11]. Although new de Bruijn graph based assembler such as EULER [12], ALLPATHS [13], Velvet [14] and ABysS [15] have attracted people’s attention, more recent assembler such as SGA [16] and fermi [17] have recalled the OLC method back to the stage benefit from increasing sequence length. As its name implies, it contains three steps:

1. The first step, overlap discovery, is the same as in the greedy approach. An all-against-all pair-wise comparison is conducted between each two short reads. In this case, however, apart from the highest-score overlap, overlaps exceeding certain threshold are stored.

2. The overlap information obtained in step 1 is utilised to construct an overlapping graph. In this graph, short reads are symbolised as nodes, and any edge connecting two nodes represents the overlap between the two reads. A transitive reduction algorithm can be applied to reduce the graph to the minimum necessary elements [18], represented in figure 6.2. Hence it works in a memory-efficient way for large genomes only.

Based on the overlap graph, the assembly work essentially is identifying paths through the graph. Ideally, a path that traverses all the nodes exactly once would indicate the desired original genome. Many issues such as repeats and errors make routing work complicated.

3. The last step is consensus computation, which integrates the fragments together according to the path abstracted from the overlap graph. Sequenced base information is required in this stage, and multiple sequence alignment is likely to be needed in a complex scenario.
Figure 6.2: A String Graph representation and its reduction

Compared with the greedy approaches, as mentioned before the main advantage of the OLC method is its capability of global analysis. As shown in figure 6.3, the greedy approach would merge two copies $X'$ and $X''$ without hesitation, and then extend the contig to a linkage of “$A - X' - X'' - C$”, discarding the fragment B. The routing analysis however can successfully add the fragment B between the two repeats.

Figure 6.3: Overlap graph of a genome containing a two-copy repeats

6.2.3 De Bruijn Graph

The de Bruijn graph approach breaks up the short reads into even shorter and unique-length segments, which is termed as k-mer [19]. The extracted k-mer data carries not only the sequence base calls, but also the degree of abundance of themselves and positions of
each $k$-mer in their original sequence. Then assembly becomes equivalent to finding a path through a de Bruijn graph constructed by the chopped $k$-mers with $k-1$ length prefixes and suffixes being the nodes. Shown in figure 6.4, length $k$ is set to be 4, so prefixes and suffixes from the $k$-mer spectrum become the nodes in graph; any edge connecting two nodes is an exact overlap between two adjacent $k-1$-mers. Because all these $k$-mers are extracted from the sequenced fragments, the path corresponding to the assembled result should cover all of them, in other words, the path should traverse all the edges. This belongs to the Eulerian Path problem, which is to find a route in a graph that visits every edge exactly once.

The De Bruijn Graph method is mostly applied for data sets composed of short reads such as ones from Solexa platforms. It holds several advantages over the other two methods: 1) It is not necessary to discover all-against-all overlap, 2) no need to store sequence and overlap data, 3) and could compress redundant sequences. There are, however, some drawbacks such as it is more vulnerable to errors and repeats.

### 6.3 Compatibility analysis for FPGA implementation

All the above mentioned algorithms are designed for general computer systems while our aim is to design an FPGA platform in real-time processing. Besides that, existing FPGA platforms focus on the alignments against an known database rather than \textit{de novo} assembly, with the exception of the de Bruijn method [20] that requires pre-processing of data and hence is not suitable for our application which needs to occur in real-time during sequencing [20-23]. Hence the potential for implementation in FPGA technique needs to be discussed. We consider the algorithms in terms of four criteria for our specific motivation: Utility in detection, post-detection assembly complexity, parallelization degree and memory requirement.
6.3.1 Utility in detection

Since we also aim to utilise the sequencing time between Nucleotide insertions in Sequencing-by-Synthesis, which is a unique requirement that has never been considered before, algorithms may have different potential for modification to be executed at the same time with the sequencing process.

Both greedy and OLC methods share a key step in the beginning, overlap computation. This computation executes the all-against-all pair-wise comparison between every two reads in the set. During the sequencing process, fragments data are not completely obtained, but it is still possible to do some comparison between sub-fragments, which may facilitate the final analysis. On the other hand, the de Bruijn graph method spends most of its time finding the Eulerian path in the complete graph, with the requirement that all the short reads should have already been sequenced. Therefore, the de Bruijn graph method is not suitable for methods that wish to utilise sequencing period.

6.3.2 Parallelisation

Overlap computation is obviously the most time-intensive component of the assembly, and the operations required might reach the level of \( C_n^2 \) in the worst case. We can see that along with the rapid increase in the number of short reads in NGS platform, the time consumed for comparison may mount up in a quadratic way. Fortunately overlap computation of every fragments in a set can be executed concurrently, which means it can be easily parallelised. As a result, both greedy and OLC methods can benefit from implementation in FPGA.

When this analysis is studied further, the OLC approach overweights its greedy counterpart even apart from their functional differences. There is larger possibility of parallelization of the whole OLC algorithm than that of the greedy, because the remaining of former algorithm (path routing) has been paralleled [24] while nothing about the latter one (greedy extension) has been reported. De Bruijn is also suitable for parallel implementation as reported in [15].

6.3.3 Post-detection assembly complexity

After the DNA detection stage, all the sensed data are obtained to finish the assembly. Assuming the planned work has been completed during the detection period, the remaining
tasks would be “path routing” for both OLC and de Bruijn, and “greedy extension” for Greedy algorithm. Path routing is a traversing algorithm such as the Chinese Postman Problem (CPP) which is solvable in polynomial time [25]. Greedy extension however is not as feasible as other two counterparts.

6.3.4 Memory requirement

As mentioned before, a string graph instead of sequence bases is constructed in OLC. Since it only takes one node and one edge to represent a fragment, it is more memory efficient. It is not the same case, however, to the de Bruijn algorithm. To overcome the short repeat problem, an equivalent transformation is introduced where each edge on the graph needs to be labelled with the read number and corresponding position [26]. Thus this $k$-mer label information would consume a lot of memory space.

6.3.5 Total Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Utility in detection</th>
<th>Post-detection complexity</th>
<th>Parallelization degree</th>
<th>Memory requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>OLC</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>de Bruijn</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

To make the comparison among these three different methods more explicitly, a list is given, from which it can be seen the OLC approach is the most suited for implementation in FPGA because its high utility of time in detection, high parallelization and medium cost in complexity and memory. Hence the OLC method is adopted to implement the FPGA system.

6.4 A Hybrid overlap searching algorithm

Due to the three distinct processes of the OLC system, the implementation can be arranged into three steps as well. As mentioned before, the overlap computation consumes most of the
time in the whole assembly process, which also means it would benefit most from parallelised in FPGA. We now discuss an efficient method for overlap computation in FPGA.

In our system the Overlap procedure is conducted on the FPGA in real-time. One of the challenges when identifying overlaps in sequences is the large scale of short reads. For instance, a DNA data set containing $N$ short reads requires $N^2$ comparisons. This can be greatly reduced as will be shown through parallelization. We proceed to describe how the overlap comparison of fragments is conducted in parallel.

6.4.1 Dynamic Suffix Comparison

To find the overlaps between sequences, approaches such as dynamic programming, filtering [27], the Burrow-Wheeler Transform [28] and the FM-index concept [29] have all been successfully applied. But these algorithms are all designed to work on the complete data set once the sensing is complete, conducting an all-against-all comparison of each fragment. In order to carry out comparison during the detection phase, a new method is required which dynamically compares each fragment every time the detection of inserted nucleotides is carried out by the sensor. So we have developed a dynamic suffix algorithm capable of dealing with comparisons of dynamically increasing fragments of DNA at the point in which they are being detected in real-time. This however requires comparison of data consisting of incomplete DNA fragments.

To solve the issue of having incomplete data, the all-against-all comparison of complete fragments is decomposed into successive window-against-window comparison phases where only certain segment of the whole sequence is used for comparison (greyed nucleotides in figure 6.5), and every time we detect a new base we execute one comparison phase. A new decomposition strategy which we call the dynamic suffix comparison approach is adopted.

During dynamic suffix comparison, each processing pixel only transmits the suffix of certain length from the detected fragment, which shifts along as new bases are detected. For example in figure 6.5a, both fragment-2 and -3 are to be compared with fragment-1. At the beginning of detection, only few nucleotide bases are available for comparison (13 in this case). Only the suffix of read-1, TCGATAA, is transmitted to other pixels, and compared with local prefixes. Both read-2 and -3 find a match between their prefix and the external suffix, as a result, they store this match as a potential overlap in their own library, termed as hitting. Pixel number 1 and offset amount 6 are recorded in Pixel 2 and 3 which indicate
the overlap information.

During the detection process, more nucleotide bases are added to reads. Overlaps may occur again with newly detected bases, so the same searching step described has to be conducted between the suffix of the detected fragment and the prefix in the other pixels; As shown in figure 6.5b, the new suffix from read-1 (TAGCCGA) is sent to other reads. Additionally, if an initial hitting was found in a previous phase, we will have enough information to re-check and confirm they are indeed genuine overlaps. By referring to the library, reads-2 and -3 have a potential overlap on read-1 with an offset amount of 6, which needs to be checked. Given the offset amount (6) and current sequence length (20), corresponding infixes are retrieved. After comparisons, shown in figure 6.5c, read-2 is shown to be a match while read-3 is a mismatch. Consequently hitting in read-2 is labelled as confirmed, but the hitting in read-3 is discarded. Thus after the whole detection process, we save only hittings which will allow us to later construct a string graph where only true overlaps exist.
6.4.2 Dynamic Programming

During the dynamic suffix comparison we must always compare the extended suffix of one pixel with the infix of another which has been identified as a hitting. However, this computation is still considerable complicated due to the errors caused during the sensing process, and the computation process is called inexact string comparison. Approaches such as q-grams [30] and spaced seeds [31] require intensive memory fetching, which has been proven as an important reason behind performance degradation in parallel computation [32].

As a result, we apply dynamic programming to find overlap matches. Dynamic programming itself calculates the edit distance\(^ 3\) of two string \(S_1\) and \(S_2\). The total process is a recursive tabular computation. If we define \(D(i, j)\) to be the edit distance of two strings \(S_1[1..i]\) and \(S_2[1..j]\), then its value can be calculated through all other \(D(m, n)\) with smaller indices expressed as:

\[
D(i, j) = \min[D(i - 1, j) + 1, D(i, j - 1) + 1, D(i - 1, j - 1) + t(i, j)]
\]  
(6.1)

where \(t(i, j)\) might be ‘1’ if \(S_1(i) \neq S_2(j)\) or ‘0’ if \(S_1(i) = S_2(j)\). When either \(i\) or \(j\) equals to one, which means there are no smaller indices, \(D(i, j)\) has to be stated explicitly through base conditions [33]:

\[
\begin{align*}
D(i, 0) &= i \\
D(0, j) &= j
\end{align*}
\]  
(6.2)

For example, a tabular computation of edit distance between strings \(AGTCTGAA\) and \(ATTCCGAAT\) is shown in figure 6.6. The table needs size of \((i + 1) \times (j + 1)\) The base conditions are highlighted by grey background. We first compute \(D(m, n)\) with the smallest \(m\) and \(n\), namely \(D(1, 1)\), according to equation (6.1). Then we fill the whole table row by row from the smallest indices. After the whole table has been solved, edit distance can be read directly from the bottom-right conner \(D(i, j)\).

6.4.3 Partitioned Dynamic Programming

Unlike the use of dynamic programming in other applications, such as finding global alignments, finding prefix-suffix matches in DNA assembly requires a considerably high consistency, i.e. above 95% similarity. Hence the final track of one dynamic programming

\(^3\)Edit Distance is the minimum steps required for transforming one string into the other by a series of edit operations on individual characters.
calculation will always be located closest to the middle diagonal line. Therefore this allows us to partition the whole calculation into serial sub-block processes, shown in Figure 6.7(a).

One direct advantage from this partitioning is that processing pixels only need to handle small amounts of data which relate to the size of the partition rather than a whole fragment, leaving the majority of the detected fragment information in the RAM. The concept of dividing a long string into smaller partitioned reads first appeared in the Four Russians’ Algorithm [34], which however calculates the results from all the sub-blocks to complete the whole dynamic programming table. In contrast our partitioned algorithm only needs to process the diagonal blocks, which further reduces timing constraints on the processing.

Given the existing DP algorithm, the pseudo-code of partition dynamic programming is given in algorithm 1. It launches a normal dynamic programming of two prefixes of two strings only, and then stores the temporary results. Afterwards the window to be compared moves to the next sub-fragment, the same calculations considering the previous results are executed. The stored information is updated every time when a computation of two sub-fragments is finished and compared with a threshold that determines the mismatch.
Algorithm 1 Partition Dynamic Programming

Require: \( S_x, S_y \) are two DNA sequences of length \( n \); sub-block length \( L \); Existing Dynamic Programming \( DP(a, b, \text{base}) \); Block shift algorithm \( \text{BlockShift(result)} \)

function \( \text{PDP}(S_x, S_y, L) \)

\[
\begin{align*}
\text{sp} & \leftarrow (1, 1); & \quad \triangleright \text{reset start point} \\
\text{while (sp.x < n && sp.y < n)} & \text{do} & \quad \triangleright \text{not the end of sequences} \\
\text{result} & \leftarrow \text{DP}(S_x[\text{sp.x} : \text{sp.x} + L], S_y[\text{sp.y} : \text{sp.y} + L], \text{base}); & \quad \triangleright \text{compare two sub-blocks starting from } \text{sp} \\
\text{score} & \leftarrow \text{score} + \text{result.edit_distance}; \\
\text{if (score} & \text{> thld_match) then} \\
& \text{flag} \leftarrow \text{mismatch}; & \quad \triangleright \text{update new bases} \\
& \text{break;} \\
\text{else} & \text{flag} \leftarrow \text{match}; \\
\text{end if} \\
\text{[shift, base]} & \leftarrow \text{BlockShift(result)}; & \quad \triangleright \text{update the starting point} \\
\text{sp} & \leftarrow \text{sp} + L + \text{shift}; \\
\text{end while} \\
\text{return flag} \\
\end{align*}
\]
Block connection

The main problem of using the partitioned dynamic programming approach is figuring out the best starting position for the next block. The starting position of the next block is determined by the results from the current block, which is often ambiguous. For example, identical numbers in the corner area of a block shown in Figure 6.7(b) indicate multiple possible starting points. Each one of these could be a possible starting point.

This problem is identical to a Hidden Markov Process with unobservable states [35]. To solve this problem, certain overlap between two neighbouring blocks is designed to give some redundancy. Note that, this redundant calculation does not correctly indicate the next start point but gives the base conditions for the next dynamic programming [36]. Therefore, we can bypass the Markov Process problem by including more information. Although this redundancy can keep the continuity between two blocks, there might still be some connection errors if the final track of one block falls outside the window of overlap defined by the level of redundancy. This is a result of not having enough base conditions for the edit distance to reach the corner of the sub-block. The probability of the existence of this situation in one block, defined as an over-shift, can be calculated using the following equation:

$$P_{\text{overshift}} = \sum_{i>r} C_L^i \cdot (1 - \epsilon_I - \epsilon_D)^{(L-i)}(\epsilon_I^i + \epsilon_D^i)$$  \hspace{1cm} (6.3)

where $r$ is the degree of redundancy, $L$ the block length, $C_L^i$ the number of unordered $i$ samples of size $L$, and $\epsilon_I/\epsilon_D$ the insertion/deletion error rate. Considering that the number of blocks is also determined by the block length, the probability of occurrence of over-shifts is found to be more dependent on redundancy rather than the block length, and therefore can be reduced by increasing the degree of redundancy.

In order to handle a situation in which blocks supply minor shifts several times and consequently the total shift amount is accumulated, we set a mechanism to allow the block to follow these shifts. The shift will only be executed when an unambiguous smallest value is found on either edge of the table. For example, in the left and middle charts in figure 6.8(a), multiple minimum values (circled) are found, and no shift action is carried out. The shifts compensation will be executed only when a clear smallest value is detected, such as in the right chart. Calculated results around the next start point (grey background) are stored as the base conditions for next block DP algorithm. Regardless of if the shifts have been implemented, the bias condition is updated according to the scores we adopt. In all these three cases, a value of 1 is added to the total score, and hence the bias values should be
subtracted by 1, shown in figure 6.8(b). The idea to compensate the block shift is explained in algorithm 2.

![Diagram](image)

(a)

(b)

Figure 6.8: shift compensation mechanism

**Algorithm 2** Block shift Compensation

Require: $result$ from dynamic programming

```plaintext
function BSC(result)
    [MinRow, IndexRow] ← min(result.row(end));  \(\triangleright\) find the minimum in last row
    [MinCol, IndexCol] ← min(result.column(end));
    if MinRow < MinCol then
        shiftRow = IndexRow;
    else if MinCol < MinRow then
        shiftCol = IndexCol;
    end if
    base ← result(shiftRow, shiftCol) − min(MinRow, MinCol);
    \(\triangleright\) update the bases
end function
```

To evaluate the performance of our partitioned dynamic programming method, we compared it with the original unpartitioned dynamic programming. We randomly generated sequences of length 400 and induced various error rates (from 1% to 5%). Different block lengths (L=5, 12, 16) and degrees of redundancy (R=1, 2, 3) are applied to investigate the reduction in accuracy according to block length and redundancy. The simulation results are shown in figure 6.9. As can be seen, increasing the degree of redundancy can significantly
improve the hitting probability (from 75% to 99.6%) whilst extending the block length has little effect, which is also consistent with equation 6.3.

What is important to note is that when over-shifts happen, the segmented dynamic programming will over-count the distances. To test the influence of over-shifts on error-rate, we investigated the scenario when the edit distance is 15, which was the case in 529 occurrences of our dynamic programming. In 520 occurrences the partition DP algorithm successfully obtained this value showing it was correctly matched 98.3% of the time. This is in agreement with equation (6.3): considering the probability of an over-shift in each sub-block, which causes errors, is $3.944 \cdot 10^{-4}$, and there are 40 sub-blocks (with an effective
length of 10) in 400-bp of sequences, the estimated error is therefore 3.944 \cdot 10^{-4} \cdot 40 = 1.58\%.
This is a small enough error to allow us to reliably use partitioned dynamic programming in our system.

We combine our dynamic suffix comparison with our portioned dynamic programming algorithm to perform a hybrid overlap searching procedure which will occur in parallel. The overlap searching process occurs as follows:

1. On every detected nucleotide we perform an exact comparison to find matching prefix and suffixes.
2. For every match found we store this information, termed as hitting, in the pixels’ local RAM.
3. Processing pixels which have been classed as having hittings continue to be compared with their matching fragments through partitioned dynamic programming. The length of fragment which is compared in this case is predetermined by the window length required for the partitioning of the dynamic programming.
4. Hittings which fail to reach a certain edit distance threshold are discarded.
5. Each processing pixel will store its overlap relationship.

Parallelisation of this procedure is realised through the implementation of multiple processing units which can convey the suffixes from one pixel to another. This is illustrated in figure 6.10 for a 3-by-3 array of processing pixels.

Each processing pixel is comprised of a local memory buffer which stores the DNA read detected for that pixel which is sent from the sensor, a comparator for detecting the common overlap with other reads, and an overlap library which stores the overlap information shown in figure 6.10(b). Detected nucleotides are encoded using 2 bits to represent the four bases \((A = 00, T = 01, C = 10, G = 11)\). To ensure that every pixel can compare its detected read with all the others, a data conveying bus passes through the array linking each pixel together and cycle data from pixel to pixel as shown in figure 6.10(a). The data from \(pixel_A\) will for example travel from \(PU_1\) (Processing Unit 1) to \(PU_2, PU_3\) and so on and so forth until \(PU_9\), which implements comparisons between data in \(pixel_A\) to all others. During the transmission of information from \(pixel_A\), all other pixels send their data to their neighbouring pixels via the data conveyor as well. This does not only save time in
comparison, but also reduces the time used for data accessing as all the pixels read their data in parallel.

6.5 FPGA Implementation

6.5.1 System Architecture

The overall system architecture is shown in figure 6.11, illustrating the different blocks used in the hierarchy. Every processing pixel is a fully functional processing unit (PU). Since these PUs receive the newly detected data from the sensor at the same time in parallel, a serial-in-parallel-out Source Buffer is applied. The data (suffix) transmission between different processing pixels is uniformly controlled by a Global Controller hence all the pixels have synchronised working phases. Their suffixes are shifted to their neighbouring pixels where a comparison takes place with the local prefixes, after which they are shifted again to the next pixel.

Due to the large data length which will be required to encode DNA fragments, detected data are stored in the RAM rather than registers. Because of the limited amount of RAM blocks available in FPGA, we have allocated 8 pixels to share one RAM space, which we cluster in a cell shown in figure 6.11. As a result, the RAM writing and reading operation...
Figure 6.11: Architecture of the proposed system
of each pixel should be scheduled well to avoid any conflicts. This job is carried out by a Memory Management Unit (MMU) in every cell.

Every pixel is comprised of four blocks: the Datapath, the Check Control, the Library and the RAM Fetch. The Datapath is responsible for the realisation of the suffix-prefix comparison and dynamic programming. Any hittings found by the datapath will be stored in the Library for further checking. These potential overlaps are monitored by the Check Control block where unqualified hitting will be dismissed. RAM Fetch block is used to connect with the MMU.

The Datapath block, includes two independent paths to an Exact Matching block and the Dynamic Programming block to carry out the suffix-prefix comparison and dynamic programming of existing hittings simultaneously. Because the infixes are required for the dynamic programming, a multiplexing mechanism is added, MUX, to choose the right data at a given time since the infix may either be located in the RAM or in a local register (Infix_Reg shown in figure 6.11). More details will be given in later subsections.

6.5.2 System Operation

![Timing chart of system indicating the processing operations](image)

Figure 6.12: Timing chart of system indicating the processing operations

Once the newly detected sensor data are ready, the global controller will firstly drive all the pixels to read their new nucleotides. In the next clock cycle, new suffixes are transmitted to neighbouring pixels, followed by a comparison between these rolling suffixes and local prefixes. Figure 6.12 shows the timing of the state machine which carries out these events, indicating this sequence of operation between the Global controller and the pixel. Every time
the state machine enters the comparison state \( Cmp \), it activates an enable signal \( Cmp_{en} \) and increments its count \( Cnt_{Cmp} \) by 1. This number not only counts the number of comparisons which have occurred, but also indicates the relative distance between any two pixels. For example, pixel \( \text{no.4} \) will compare its prefix with the suffix from pixel \( \text{no.3} \) when \( Cnt_{Cmp} \) equals to 1, then compare with pixel \( \text{no.2} \) when \( Cnt_{Cmp} \) equals to 2 and so on.

RAM reading has to be done in two phases, \( Rd_1 \) and \( Rd_2 \), because the infix maybe located anywhere between two consecutive words. Also as shown in figure 6.12, both the RAM reading and checking of subsequent infixes can be executed concurrently, which makes it more time efficient. The checked results are updated when the dynamic programming calculation finishes.

### 6.5.3 Memory structure and Infix retrieval

Detected sequence data in each pixel is segmented and stored in two locations, a local register which always contains the pixels suffix and a RAM which stores the subsequent bases forming the infix. This is shown in figure 6.13(a) and (b) whereby the local suffix register is 16 bases spanning two suffixes in length, and is stored in the RAM as segmented words of 8 bases (16bits) in length.

Since the overlap offset between two fragments being compared can be arbitrary in length as shown in figure 6.13(a) and (b), the retrieval of the infix which is being compared needs consideration as this may either lie in the suffix register or in the RAM. Assuming the suffix which has the same length as every word to be \( L_{xfix} \) and sequences overlap with an offset \( L_{offset} \), the infix will always be located in the suffix register for comparison when the offset of two overlap reads \( L_{offset} \) is no larger than the length of a single suffix \( L_{xfix} \) \((L_{offset} \leq L_{xfix})\), shown in figure 6.13(a).

In the alternative case where the infix needs to read from the RAM, since data in RAM are processed in terms of minimum word length, infix retrieval always needs two RAM reading operations as infixes may be located anywhere within two consecutive words. For example, shown in figure 6.13(b), there is a 10-base offset between these overlaps, and current length is 28, causing the corresponding infix to be located between both the second \((\text{word1})\) and third words \((\text{word2})\).
6.5.4 Hitting Library

When a hitting is found between two fragments, the necessary information detailing the relationship between them and information on how subsequent checking using the portioned dynamic programming is combined and stored in a local library. The hitting library for each match is composed of four kinds of data:

- **Offset**: This indicates where the overlap takes place between the two compared fragments.
- **Distance**: This number indicates the relative distance in terms of pixel number between two pixels which have overlapping reads, with which the number of the original overlapping pixel can be calculated.
- **Score**: This indicates the accumulated edit distance so far for the dynamic programming.
- **Base**: These are the base condition numbers used for the next partitioned dynamic programming.

The library is implemented using a FIFO, as shown in figure 6.14. The seed at the bottom “Seed_0” is always checked first. If an overlap continues to be confirmed as a hitting it is re-added back to the top of the library, otherwise it is discarded and thus deleted.
When the number of DNA fragments and thus pixel number is drastically increased, the size of the library storing the seeds becomes an issue. Firstly more bits are needed to store the rotation number (pixel number) due to an increase in the number of pixels. Another problem arises from the fact that it tends to find false-positive results since there is more repeated data. The library should at least be capable of storing these false-positive hitings until their checking phases where they can be discarded. The mean number of random seeds due to false positive hitings can be calculated as:

\[ N_{\text{ran}} = \left( \frac{1}{4} \right)^L \cdot N_{\text{pixel}} \]  

(6.4)

where \( N_{\text{ran}} \) is the number of random matches, \( L \) is the length of the blocks, and \( N_{\text{pixel}} \) is the number of pixels. Evaluating this implies a length of more than 10 is necessary for large scale systems to compress random hitings and hence avoid using a very large library.

### 6.5.5 Implementation of Dynamic Programming in Hardware

The hardware implementation of dynamic programming can be efficiently implemented using a systolic array. The systolic array instantiates the hardware for computational cells on a diagonal wave-front [37] as shown in figure 6.15(a). Therefore only the arithmetic unit (AU) which exists on the diagonal wave-front \( \text{wave}_0 \) is required. The results of wave-fronts \( \text{wave}_{N-1} \) and \( \text{wave}_{N-2} \) become the input base conditions for the calculation at \( \text{Wave}_N \). As a result, the computation time is linearly dependant to the number of waves.

Since the error rate is expected to be less than a certain value, e.g. 1.2% [26], the error
occurrence in each block should therefore be kept within a certain bound as well to avoid a large error rate. To achieve this an upper bound of calculated scores can be applied on every block. For example in figure 6.15(b), the redundancy degree is set to 2 and the regional maximum score to 3, hence the width of the wave-front can also be constrained within 5 bits. This upper bound limit dramatically reduces the utilisation of FPGA resources, not only by reducing the processing units needed but also simplifying the circuits in each unit as the maximum number is limited.

Figure 6.16: Hardware Implementation of Dynamic Programming

The system used for the hardware implementation of dynamic programming is shown in figure 6.16. Although it is possible to reuse the AUs and reduce the number of comparison to
the registers, the combinational multiplexing circuits required for this make it impractical. Hence we fit an AU to every register as shown in figure 6.16. By doing so every AU has fixed inputs. Also because of the movement of wave-front, the character inputs to each AU are varying. To further simplify our circuit, we let the input strings shift in opposite directions, where String/1/ stands for the first character of the string.

6.5.6 Hardware Resource

The prototype system is implemented in an Altera Stratix IV FPGA. To demonstrate the system a 64 cell array is synthesised, with every cell containing 8 pixels, resulting in a 512 pixel array. The length of prefixes, infixes and suffixes are set to be 8. Accordingly, each word occupies 16 bits as a nucleotide occupies 2 bits. Considering the large amount of information required for the library seeds, the library FIFOs are set to be 32 bits, where Offset, Distance, Score and Bases use 11, 7, 4 and 10 bits respectively. The redundancy degree is set to 2 and the maximum score in every block is set 3. The critical parameters are summarised in Table 6.2.

Table 6.2: Parameter settings for FPGA Implementation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell number</td>
<td>64</td>
</tr>
<tr>
<td>Pixel number</td>
<td>8</td>
</tr>
<tr>
<td>Window length</td>
<td>8</td>
</tr>
<tr>
<td>RAM width</td>
<td>16</td>
</tr>
<tr>
<td>FIFO width</td>
<td>32</td>
</tr>
<tr>
<td>Redundancy</td>
<td>2</td>
</tr>
<tr>
<td>Maximum score</td>
<td>3</td>
</tr>
</tbody>
</table>

The resource usage of the FPGA is summarised in Table 6.3 where C-Luts stands for the Combinational LUTs and M-Luts the Memory LUTs. Maximum frequency is optimised to 211.7MHz.

Table 6.3: FPGA Resource Usage

<table>
<thead>
<tr>
<th>Type</th>
<th>C-Luts</th>
<th>M-Luts</th>
<th>Register</th>
<th>Memory</th>
<th>fmax(Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bits</td>
<td>bits</td>
<td>bits</td>
<td>bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>298,821</td>
<td>16,384</td>
<td>249,088</td>
<td>917,504</td>
<td>211.7M</td>
</tr>
<tr>
<td></td>
<td>(70%)</td>
<td>(8%)</td>
<td>(59%)</td>
<td>(4%)</td>
<td></td>
</tr>
</tbody>
</table>
6.6 Performance Evaluation

By applying multiple processing units operating in parallel, the computation complexity has been decreased from $O(n^2)$ to $O(n^2/m)$ where $n$ stands for the number of short reads and $m$ the number of processing pixels. The speed of comparing individual bases is determined by both the degree of parallelization and the clock frequency which can be calculated as:

\[
\text{Comparison rate} = \# \text{Parallel pixels} \times \text{Frequency}/2
\]
\[
= 512 \times 200MHz/2
\]
\[
= 51.2 \text{ Giga comparison per Second}
\]

where the frequency is divided by 2 to take into account the time used for data transportation. Assuming every nucleotide detection takes four seconds during DNA sequencing, our prototype system is capable of accommodating the comparison of $4 \times 51.2G$ bases in the detection period, which results in a capability of accommodating around $452K$ ($\sqrt{4 \times 51.2G}$) short reads in real-time.

Although the absolute speed we estimated cannot beat the state-of-art PC-based counterparts [38], we argue that the proposed platform is capable of utilising the detection time and hence achieving the comparison results immediately after completing of detection. Whereas existing PC based algorithms process the complete data after the detection has finished, so they would effectively take longer for the whole process of DNA sequencing. The same situation applies to current FPGA acceleration methods where no real-time comparison can be executed [39].

To estimate the advantage of our method, we compare the assembly of E.coli DNA with 10x coverage against existing method using SSAKE. When the input data set contains 400k short reads with the length of approximately 100bp causing a 40Mbp input amount (e.g. E.coli with 10X coverage), it takes around 1.1 hours to detect. Assembling these data using SSAKE would take a further 2.5 hours [38], whereas our system finishes immediately after detection because both the transitive reduction algorithm and routing algorithm such as Chinese Postman algorithm can be executed within polynomial time [18], illustrated in figure 6.17.
6.7 DNA Assembly

We evaluated the accuracy of our system on three real DNA data sets of bacterial genomes, PhiX-174, UW4 and sp.GeD10, of different sequence length and coverage, the details of which are shown in table 6.4. The genomes were fragmented into 512 short reads of 200bp in length with a 3% error induced to reflect a real detection scenario. We then assembled these fragments back in Matlab and compare the re-generated DNA sequence with the original one.

As can be seen in table 6.4, the similarities of the assembled and original sequences are more than 96% which is very close to the detection error rate (3%). In addition to that, we also evaluated the system for the ideal error-less scenario, where it managed to restore the original sequence with 100% accuracy. Hence, the comparison platform can successfully assemble the detected DNA with virtually no error.

Table 6.4: Overlap hitting found by proposed algorithm for three bacterial genomes

<table>
<thead>
<tr>
<th>Species</th>
<th>Bank Ref.</th>
<th>Sequence Length</th>
<th>Coverage</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhiX-174</td>
<td>J02482.1</td>
<td>5386bp</td>
<td>19X</td>
<td>96.15%</td>
</tr>
<tr>
<td>UW4</td>
<td>NZ_AQVS08757bp</td>
<td>5984bp</td>
<td>11.7X</td>
<td>96.49%</td>
</tr>
<tr>
<td>sp.GeD10</td>
<td>CAVI01</td>
<td>9688bp</td>
<td>10.6X</td>
<td>96.19%</td>
</tr>
</tbody>
</table>

Figure 6.17: Timing diagram showing the parallel assembly process with detection
6.8 Conclusion

This chapter presented a system which is capable of utilising the actual detection time during DNA sequencing to accelerate the assembly process through a novel parallelised architecture on FPGA. The Overlap-Layout-Consensus (OLC) method was found to be the most suitable method for this parallel implementation. Due to the necessity of an all-against-all comparison in this method, which is quite time-consuming, a hybrid overlap searching algorithm was proposed which can quickly find hittings and carefully check their validity. The checking process was carried out through dynamic programming. To make the system scalable for an increasing data set, the normal dynamic programming (DP) algorithm was modified into a partitioned DP algorithm which allowed the comparison to finish in discrete phases. The partitioned DP algorithm was proven to be valid for small error rate sequence comparison.

The system was then implemented in a hierarchical structure for an FPGA implementation. Every processing unit contained the datapath responsible for arithmetic computation, library storing the seeds and check control unit that monitors the library. Because of the limited slice numbers, several PUs were required to share one RAM block in a cell, whose read and write operations are scheduled by a memory management unit. All the cells were controlled by a global controller achieving synchronised operation.

Due to this hierarchical architecture, the maximum operating frequency was optimised to 211 MHz. With a realisation of 512 PUs, comparison rate of 51.2G/s is achievable. This comparison rate allows the system to process the data from a detection array with 500k sensors in real-time. Its utilisation of the detection period makes it a promising platform as it can be run in parallel during the detection phase of DNA sequencing and allow instantaneous assembly of the fragments of DNA as soon as the detection is complete. This ultimately shortens the time required for DNA sequencing when used with NGS sensing technologies such as semiconductor based sequencers.

In terms of large genome assembly applications, the remaining bottleneck would be the data transfer rate between the FPGA and workstations. Since the limited memory space in FPGA boards, the majority of fragments data would be stored in the central memory of a workstation, then the data has to be send to FPGA to execute the comparison. Besides, increasing random prefix matching would also exert a resource pressure to the FPGA.
Bibliography


Chapter 7

Conclusion

This thesis explores and develops new methods to allow faster and more efficient DNA sequencing and assembly using semiconductors. DNA sequence detection efficiency can be improved by overcoming existing drawbacks in the sensing front-ends, and the assembly process will be faster if the detection time can be utilised. Therefore this research focused on the potential of sensor improvement and real-time assembly algorithms. It involves three main novel contributions: ISFETs arrays in CMOS utilising capacitive feedback, automatic gain calibration system and real-time DNA comparison implementation using an FPGA.

Chapter 2 first introduces the fundamental features of DNA and the principles of base pair incorporation. Because the incorporation process can only take place between specific pairs and releases the phosphate group as well as hydrogen ions, various sequencing approaches can be realized for detection depending on which materials they are sensitive to. Several commercial sequencing platforms are introduced then, among which the semiconductor based approach is found to be the most promising in terms of speed, size, cost and integration of signal processing units. Then the theory behind chemical sensors using semiconductors (ISFETs) is detailed. The function of ISFETs implemented in unmodified CMOS process (Extended-gate ISFETs) are described. The challenges of these Extended-gate ISFETs for low cost fabrication process such as trapped charge, sensitivity loss and drift which hamper them from being applied in large arrays are also discussed. Finally the targets specifications of our new front-ends are defined to overcome these existing problems.
Chapter 3 focuses on novel ISFET front-ends. A new design methodology is proposed which applies the capacitive feedback structure. Because of this feedback, the potential of the four terminals of the ISFET transistor are all locked to perform a constant-voltage-constant-current (CVCC) circuit. This CVCC circuit bootstraps the parasitic gate-source capacitor $C_{gs}$ and consequently solves the sensitivity loss problem. Because the gate voltage has already been locked, a low-leakage switch can be implemented to reset the gate voltage. Due to the reset phase, issues due to trapped charge and drift are also resolved. To further decrease the power consumption, a single-stage front-end is also proposed which trades off linearity and accuracy. Afterwards an extreme case to form a miniature pixel is investigated where parasitic capacitor is utilised as feedback capacitor. Tests results show that all three types of front-ends reach our expected functionality such as low-leakage and in-pixel gain. Different sensitivities were found for these front-ends, but may have come from the different fabrication process. Finally a comparison between them was given.

Chapter 4 proposes an automatic gain calibration system that could compensate the gain deviation of pixels in the ISFET array. Since this deviation is caused by the process mismatch of both passivation and feedback capacitors which is uncontrollable, a closed loop feedback calibration system is required. Because the chemical signal is located in the low frequency spectrum, the high frequency band can be utilised for calibration. As a result, a high frequency sine wave is superimposed on the chemical signal via the reference electrode. The sensor pixel should have the same gain for both chemical and sine wave signals as long as they are in the bandwidth. By examining the amplitude of high frequency sine wave, the gain can be calculated and calibrated accordingly. To achieve consistent settling time for all the pixels, a linear-in-dB response was developed by approximating an exponential relationship. The amplitude is detected by a rectifier followed by a lowpass filter. The second order harmonic from the rectifier however deteriorates the calibration accuracy. The original system was found to have a trade-off between settling time and accuracy, which can be bypassed by adding additional lowpass filters. Hence, a total third order lowpass system is implemented. Test results represent a good harmonic suppression without sacrificing settling time. Nevertheless, the final accuracy is also under the influence of input-referred noise both from the sine wave and system itself.

Chapter 5 combines the sensor array and calibration system together to present a novel $32 \times 32$ ISFET sensing array. A digital correlated double sampling system is applied to solve
the offset problem, which is challenging to implement in the analogue domain due to the medium-& long-term monitoring period. An SPI protocol is used to send the digitalised data to off-chip memory, as well as data retrieval. The offset cancellation is tested to be functional. However leakage currents from the switch still make the sensor inaccurate. Subtraction of leakage current could improve the final results but require post signal processing. The major problem we met was that the gain distribution of pixels in the array has little standard deviation, and the effect of calibration system is less apparent in the small array presented. Nevertheless the array achieves good sensor matching alleviating problems of trapped charge and offset in ISFET sensors.

Chapter 6 describes a real-time DNA fragments comparison system implemented in FPGA. The Overlap-Layout-Consensus (OLC) method is found to be most suitable for utilising sequencing time and parallelisation. To handle the incomplete data set during the sequencing time, a novel hybrid comparison algorithm is proposed. The original all-against-all comparison step in OLC method is firstly decomposed into successive window-against-window comparison phases, and then dynamic programming is adopted on the exact comparison to achieve high speed but error-tolerant computation. Hierarchical implementation in FPGA is represented where processing units are paralleled and controlled by one global controller. The validation of the proposed system was proven by the assembly of three real DNA sets even with deliberate errors introduced.

7.2 Recommendations for Future Work

7.2.1 Optimisation and development in front-ends

As can be seen in the tested results, leakage from the sensors becomes the main problem of the proposed system. The author tried to avoid the use of switches by utilising the small leakage current between the two metal layers in the chip which did not give positive results. However there are publications representing these leakage effects and if future research could characterise it, we can utilise this feature to reset the DC output which is similar to a highpass filter with an extreme low cut-off frequency. Additionally, if this comes to be true, the 3-T front-ends can be widely used.
7.2.2 Potential in calibration system

Due to the existence of the clock, a digital calibration system is an appealing approach to compensate gain deviation in the array. The digital system could be based on a sample-&-hold methodology to detect the amplitude, which is very sensitive to noise if high resolution is required. Future work could be the integration of waveform generator to obtain clean high frequency wave and the investigation of square wave usage which could relax the bandwidth requirement for front-ends. Calibration functionality can be done via a digital tuned variable gain amplifier where the successive approximation approach is preferred to set the tuning factor.

To further prove the functionality of the AGC, it is recommended to combine it with sensor array which has significant gain deviation, for example, an array comprising 3-T front-ends.

7.2.3 Fully integrated DNA sequencing & assembly system

In terms of the DNA fragments comparison work, future work could focus on the connection between the FPGA and desktop. Because the number of processing units is always smaller than that of DNA short reads, it is necessary for each processing unit to serve multiple short reads. Besides, it is impossible to store all the DNA data on the FPGA, fast data transferring between desktop and FPGA is crucial for large size genome comparison.

Finally, there can be a system combining the ISFET sensing array, sequence comparison platform in FPGA and a workstation all together. With that system the sensing and comparison can be executed concurrently with real-time communication, and at the end of detection, assembly process can be finished immediately.

7.3 Personal perspective

In the previous chapters, a novel front-end was proposed to handle the non-ideal effects such as trapped charge, capacitive division and drift that exist when fabricating ISFETs in unmodified CMOS. Although these problems have been well resolved, some new issues still challenge the system like leakage current and gain mismatch. If we have a look at existing commercial approaches, we see that none of them have tried to solve the problems using
circuits but overcome it through post-processing. The common method is to etch the original passivation and then glaze another sensing layer, which can create a much larger passivation capacitor. This reduces attenuation, removes trapped charge effects and improves sensitivity. Another argument in favour of post-processing is that in order to allocate the DNA beads precisely above the sensors, there has to be additional fabrication steps to build a physical well.

ISFET fabricated using unmodified CMOS can still be advantageous however for reducing the cost of developing sensing systems with the trade-off as demonstrated in this work. Despite limitations, there is still great potential for creating smaller scale sensing arrays for diagnostics that can benefit from the approaches developed in this work.
Appendix A

Publications arising from this work

- Hu, Y.; Georgiou, P., "A real-time de novo DNA sequencing assembly platform based on an FPGA implementation," Computational Biology and Bioinformatics, IEEE/ACM Transactions on


- Yuanqi Hu; Georgiou, P., "3-T ISFET front-end utilising parasitic device capacitance," Electronics Letters, vol.50, no.21, pp.1507,1509, October 9 2014


Appendix B

Fabricated chips

Figure B.1: Dolphin – 32x32 ISFET array system
Figure B.2: Panda – 32x32 ISFET array system

Figure B.3: Altair – ISFET characterisation devices and Automatic Gain Control System

Figure B.4: Gonzo – ISFET characterisation devices
Figure B.5: Kimi – ISFET characterisation devices

Figure B.6: Kermit – ISFET characterisation devices
Appendix C

DNA Detection in CMOS

Firstly it is necessary to mention that detecting a DNA is not necessary to be limited in chemical approaches. The hybridization of a DNA target will cause changes in mass, viscosity and surface stress etc. By monitoring these physical changes DNA sensors can also be realized. All the cantilever-based and resonance-based approaches fall into this category [1-5]. As figure C.1(a) shows, the hybridization of DNA will add additional weight to the piezoelectric layer, which can be detected in various ways. Figure C.1(b) presents the cantilever-based approaches, where both static bending and resonant frequency can be sensed.

![Diagram](image)

Figure C.1: Demonstration of two DNA detection methods relying on physical changes

Chemical or charge sensing is another common way used for DNA detection. A immo-
bilized DNA strands on the electrode could be approximately model as equivalent circuits shown in figure C.2 [6, 7]:

![Figure C.2: The approximation model of DNA strands attached on the electrode](image)

where $R_s$ depends on the interface-solution property, $R_p$ describes the insulation degree of interface can could be neglected for well-formed layers. $C_p$ is determined by the physical and chemical characteristics of the bio-layer attached on the surface, composed of DNA strands.

When the hybridization takes place and DNA duplex is formed, a variation of electrical property of the solution can be utilized for the detection. In [8-10] it was shown how to measure the conductance or resistance changes and other work such as [11-13] monitors the capacitance fluctuation.

The mentioned above works, however, could only detect the hybridization of designated probes, which limits the application to DNA resequencing. The semiconductor de novo sequencing techniques still reply on sensing the electrons released from Nucleotide incorporation explained above. In [14] it was shown that the transient current on electrode caused by charge movement could be measured. But the majority of publishes focus on the potential changes on the surface where field effect is the most effective method to monitor.
Bibliography


Appendix D

Codes of Fragments Comparison algorithm for FPGA

The hierarchy of entity instantiation of the comparison platform is shown in figure D.1.

Figure D.1: The hierarchy of entity instantiation
The code for each entity is given below:

Listing D.1: System

```vhdl
-- FILE name: SYSTEM.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 01/05/2012
-- version: 1.0
-- Abstract: The whole comparison system comprises a pixel array, a global controller governing the array, a MMU and a RAM.
-- Called by: SYSTEM.vhd
-- Revision history: N/A

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;
use WORK.all;

entity SYSTEM is
port(
    Clk, Rst: in STD_LOGIC;
    Rd_Lib_req: in STD_LOGIC;
    Rd_Lib_ack: out STD_LOGIC;
    Rd_FIFO_en: in STD_LOGIC;
    Overlap_o: out STD_LOGIC_VECTOR(31 downto 0)
);
end SYSTEM;

architecture STRUCT of SYSTEM is
component CELL port
    Clk, Rst: in STD_LOGIC;
    Data_in: in DATA_ARRAY(NUM_PIXEL-1 downto 0);
    Suffix_Reg_en: in STD_LOGIC;
    Prefix_en: in STD_LOGIC;
    Suffix_tran_en: in STD_LOGIC;
    Tran_ing: in STD_LOGIC;
    Comp.en: in STD_LOGIC;
    Cnt.Rd: in INT_RD;
    Cnt.Comp: in INT_PIXEL_ALL;
    Wr_RAM_Req: in STD_LOGIC;
    Rd_Lib_Req: in STD_LOGIC;
    Rd_Lib_ack: out STD_LOGIC;
    Wr2FIFO_en: out STD_LOGIC;
    Overlap2FIFO: out OVERLAP;
    Suffix_i: in SEQ_DNA(LENG_FIX-1 downto 0);
    Suffix_o: out SEQ_DNA(LENG_FIX-1 downto 0);
end component;

type SUFFIX_CELL is array (NATURAL range <>) of SEQ_DNA(LENG_FIX-1 downto 0);

signal Wr_RAM_Req: STD_LOGIC;
signal Suffix_Reg_en: STD_LOGIC;
signal Prefix.en: STD_LOGIC;
signal Suffix_tran_en: STD_LOGIC;
signal Tran.ing: STD_LOGIC;
signal Comp.en: STD_LOGIC;
signal Cnt_Rd: INT_RD;
signal Cnt.Comp: INT_PIXEL_ALL;
signal Rd.en: STD_LOGIC;
signal Rd.ack: STD_LOGIC;
signal Data_Cell: DATA_CELL(NUM_CELL-1 downto 0);

signal Rd.Lib_req_Vec: UNSIGNED(NUM_CELL-1 downto 0);
```

---
signal Rd_Lib_ack_Vec: UNSIGNED(NUM_CELL-1 downto 0);
signal Wr2FIFO_Vec: UNSIGNED(NUM_CELL-1 downto 0);
signal Overlap_Vec: OVERLAP_VEC(NUM_CELL-1 downto 0);

begin
CELL_CLUSTER: for i in 0 to NUM_CELL-1 generate
    CELL_INST: CELL port map(
        Clk=>Clk, Rst=>Rst,
        Data_in=>Data_Cell(i),
        ------------ signals connecting with global controller -------------
        Suffix_Reg_en=>Suffix_Reg_en,
        Prefix_en=>Prefix_en,
        Suffix_tran_en=>Suffix_tran_en,
        Tran_ing=>Tran_ing,
        Comp.eno=>Comp.en,
       Cnt_Rd=>Cnt_Rd,
        Cnt.Comp=>Cnt.Comp,
        Wr_RAM_Req=>Wr_RAM_Req,
        ------------ signals connecting between clusters -----------
        Suffix_i=>suffixes(i),
        Suffix_o=>suffixes((i+1) mod NUM_CELL),
        -- read out seeds data in FIFO
        Rd_Lib_req=>Rd_Lib_req_Vec(i),
        Rd_Lib_ack=>Rd_Lib_ack_Vec(i),
        Wr2FIFO_en=>Wr2FIFO_Vec(i),
        Overlap2FIFO=>Overlap_Vec(i));
    end generate CELL_CLUSTER;

-------------------- instantiation of global controller --------------------------
GLO_CTRL_INST: entity GLOBAL_CTRL port map(
    Clk=>Clk, Rst=>Rst,
    -- ports connecting with the source
    Rd.en=>Rd_en,
    Rd.ack=>Rd.ack,
    -- ports connecting with the local controll unit
    Cnt_Rd=>Cnt_Rd,
    Cnt.Comp=>Cnt.Comp,
    -- ports connecting to MMU
    Wr_RAM_Req=>Wr_RAM_Req,
    -- ports connecting with datapath
    Suffix.Reg.en=>Suffix.Reg.en,
    Prefix.en=>Prefix.en,
    Suffix_tran=>Suffix_tran,
    Tran.ing=>Tran.ing,
    Comp.eno=>Comp.en);

-------------------- instantiation of Library fetching management unit ------------------
LIB_FETCH_INST: entity LIB_FETCH port map(
    Clk=>Clk, Rst=>Rst,
    ------------ ports connecting with global controller -------------
    Rd.FIFO.Glo.en=>Rd.FIFO.en,
    Rd.Lib.Req=>Rd.Lib.Req,
    Rd.Lib.ack=>Rd.Lib.ack,
    Overlap.en=>Overlap.o,
    ------------ ports connecting with Cells -----------
    Rd.FIFO.req.veo=>Rd.Lib.req.Vec,
    Rd.FIFO.ack.veo=>Rd.Lib.ack.Vec,
    Wr2FIFO.Vec=>Wr2FIFO_Vec,
    Overlap.Vec=>Overlap.Vec);

SOURCE_INST: entity SOURCE PORT map(
    Clk=>Clk, Rst=>Rst,
    -- ports for sending data to comparator
    Rd.ack=>Rd.ack,
    Rd.en=>Rd.en,
    Data_Cell=>Data_Cell);
end STRUCT;

Listing D.2: Library Fetch Management

--- FILE name: SYSTEM.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 01/05/2012
-- version: 1.0
-- Abstract: The whole comparison system comprises a pixel array, a global controller governing the array, a MMU and a RAM.
-- Called by: SYSTEM.vhd
-- Revision history: N/A
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;
use WORK.all;

entity LIB_FETCH is
  port ( 
    Clk, Rst: in STD_LOGIC;
    Rd_FIFO_GLO_en: in STD_LOGIC;
    Rd_Lib_Req: in STD_LOGIC;
    Rd_Lib_ack: out STD_LOGIC;
    Overlap_o: out STD_LOGIC_VECTOR(31 downto 0);
  );
architecture STRUC of LIB_FETCH is
begin
  FIFO_MANAGE_INST: entity FIFO_MANAGE port map ( 
    Clk=>Clk, Rst=>Rst,
    Rd_Lib_Req=>Rd_Lib_Req,
    Rd_Lib_ack=>Rd_Lib_ack,
    Overlap_o=>overlap
  );
  FIFO_ALL_INST: entity FIFO_ALL PORT map ( 
    clock=>Clk, 
    data=>overlap, 
    rdreq=>Rd_FIFO_GLO_en, 
    wrreq=>Wr2FIFO_en, 
    q=>Overlap_o, 
    usedw=> usedw
  );
end STRUC;

-- FILE name: GLOBAL_CTRL.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 01/05/2012
-- version: 2.0
-- Abstract: A global controller controls all the transmission, comparison and check processes in all pixels. The above mentioned
-- steps are executed simultaneously in each pixel. The controlling procedure is triggered by an external
-- source. This
-- global controller communicates with local controllers, local datapaths and MMU. In this version, suffix
-- data are
-- transmitted serially, consuming LANG_FIX clocks, and data comparison can be finished in one clock period.
-- Some abbreviations: Comp=compare, ext=external, int=internal, Mas=master, Req=requests, Sour=source, Trans=
-- Called by: SYSTEM.vhd
-- Revision history: 23/08/11-v1.0
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;

entity GLOBAL_CTRL is
  port(
    Clk, Rst: in STD_LOGIC;
    -- ports connecting with the source
    Rd_en: in STD_LOGIC;
    Rd_ack: out STD_LOGIC;
    -- ports connecting with the local control unit
    Cnt_Rd_o: out INT_RD;
    Cnt_Comp_o: out INT_PIXEL_ALL;
    -- ports connecting to MMU
    Wr_RAM_Req: out STD_LOGIC;
    -- ports connecting with datapath
    Suffix.Reg.en: out STD_LOGIC;
    Prefix.en: out STD_LOGIC;
    Suffix_trans.en: out STD_LOGIC;
    Tran_ing: out STD_LOGIC;
    -- '1'=compare the unknown reads; '0'=check the existing overlap
    Comp.en: out STD_LOGIC
  );
end GLOBAL_CTRL;

architecture RTL of GLOBAL_CTRL is
  type MULTI_STATE is (IDLE, RD, TRAN, COMP, WAIT_SOUR);
  signal ss, ss_next: MULTI_STATE;
  signal Cnt_Rd: INT_RD;
  signal Cnt_Comp: INT_PIXEL_ALL;
  signal Comp_en_LN2, Comp_en_LN1: STD_LOGIC;
begin
  SS_PROC: process(Clk, Rst)
  begin
    if Rst='1' then
      ss<=IDLE;
    elsif rising_edge(Clk) then
      ss<=ss_next;
    end if;
  end process SS_PROC;
  -- calculate the next state for dataflow control
  SS_NEXT_PROC: process(ss, Rd_en, Cnt_Rd, Cnt_Comp)
  begin
    ss_next<=ss;
    case ss is
      when IDLE =>
        if Rd_en='1' then -- enable signal from the external source
          ss.next<=RD;
          end if;
        when RD =>
          if Cnt_Rd>=LENG_FIX-1 then -- start to transmit the suffix among pixels when Cnt=LENG_FIX-1
            ss.next<=TRAN; -- then after this read, we will have LENG_FIX bases.
          else
            ss.next<=WAIT_SOUR;
          end if;
        when TRAN =>
          if Cnt_Comp=NUM_PIXEL*NUM_CELL-1 then -- all the comparisons have been launched
            ss.next<=WAIT_SOUR; -- do the check phase
          else
            ss.next<=COMP; -- there are still some pixel haven't been compared
          end if;
        when COMP =>
          ss.next<=TRAN; -- to get a new suffix from another pixel
        when WAIT_SOUR =>
          if Rd_en='0' then
            ss.next<=IDLE;
          end if;
        when others =>
          null;
    end case;
  end process SS_NEXT_PROC;

  DATA_PROC: process(Clk, Rst)
  begin
    if Rst='1' then
      ------------------------ reset the output ports ----------------------
      -- signal connecting to peripheral circuits
      Rd_ack<='0';
      Comp_en<='0';
Tran_ing<='0';
Wr_RAM_Req<='0'; -- to MMU
-- signal connecting to datapath
Suffix.Reg_en<='0';
Suffix.trans.en<='0';
Prefix.en<='0';
-- reset the internal signals
Cnt.Rd<='0';
Cnt.Comp<='0';
Comp.en.LN2<='0';
Comp.en.LNI<='0';

elsif rising_edge(Clk) then
  case ss is
    when IDLE =>
      Cnt.Comp<='0';
      Comp.en<='0';
      Suffix.Reg_en<='0';
      Suffix.trans.en<='0';
      Prefix.en<='0';
      Rd.ack<='0';
    when RD =>
      -- set the enable signal to Datapath
      Suffix.Reg_en<='1';
      if Cnt.Rd<=LEN.FIX then
        Prefix.en<='1';
        -- build the prefix of every pixel
        end if;
      -- signals to other parts
      Rd.ack<='1';
      -- to sensor source
      Cnt.Rd<=Cnt.Rd+1;
      Tran.ing<='1';
      if (Cnt.Rd mod (WIDTH.WORD/2))=(WIDTH.WORD/2)-1 then
        -- every time there are data of a word
        Wr_RAM_Req<='1';
        -- transfer the data from register to RAM
        end if;
    when TRAN =>
      -- reset the enable signals
      if Cnt.Comp=NUM.PIXEL-1 then
        Wr_RAM_Req<='0';
        -- end if;
      Suffix.Reg_en<='0';
      Prefix.en<='0';
      -- set the signal used in this state
      Tran.ing<='1';
      Suffix.trans.en<='1';
    when COMP =>
      -- clear the signals used in last state - TRAN
      Suffix.trans.en<='0';
      -- enable the comparison
      if Cnt.Rd=OVLAP.TH then
        Comp.en<='1';
        end if;
      Cnt.Comp<=Cnt.Comp+1;
    when WAIT.SOUR =>
      Cnt.Comp<='0';
      Suffix.trans.en<='0'; -- stop the transmission
      Tran.ing<='0';
      Comp.en.LN2<='0';
      Comp.en.LNI<='Comp.en.LN2';
      if Comp.en.LNI='1' then
        Comp.en<='1';
        else
        Comp.en<='0';
        end if;
      Suffix.Reg_en<='0';
      Prefix.en<='0';
    when others =>
      null;
  end case;
  end if;
end process DATA_PROC;
-- mapping the output
Cnt.Rd.o<=Cnt.Rd;
Cnt.Comp.o<=Cnt.Comp;
end RTL;
```
-- FILE name: SYSTEM.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 01/05/2012
-- version: 1.0
-- Abstract: The whole comparison system comprises a pixel array, a global controller governing the array, a MMU and a RAM.
-- Called by: SYSTEM.vhd
-- Revision history: N/A

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;
use WORK.all;

entity CELL is
    port(
        Clk, Rst: in STD_LOGIC;
        Data_in: in DATA_ARRAY(NUM_PIXEL-1 downto 0);
        Suffix_Reg_en: in STD_LOGIC;
        Prefix_en: in STD_LOGIC;
        Suffix_tran_en: in STD_LOGIC;
        Tran_ing: in STD_LOGIC;
        Comp_en: in STD_LOGIC;
        Cnt_Rd: in INT_RD;
        Cnt_Comp: in INT_PIXEL_ALL;
        Wr_RAM_Req: in STD_LOGIC;
        ------------ signals connecting with global controller ------------
        Suffix_i: in SEQ_DNA(LENG_FIX-1 downto 0);
        Suffix_o: out SEQ_DNA(LENG_FIX-1 downto 0);
        ---------------- read out seeds data in FIFO ----------------
        Rd_Lib_req: in STD_LOGIC;
        Rd_Lib_ack: out STD_LOGIC;
        Wr2FIFO_en: out STD_LOGIC;
        Overlap2FIFO: out OVERLAP;
        ------------ signals connecting between clusters ----------
        Cnt_Rd_en: in STD_LOGIC;
        Cnt_Comp_en: in STD_LOGIC;
        Wr_RAM_en: in STD_LOGIC;
        usedw_fifo: STD_LOGIC_VECTOR (9 DOWNTO 0);
        data: STD_LOGIC_VECTOR(WIDTH_WORD-1 downto 0);
        q: STD_LOGIC_VECTOR(WIDTH_WORD-1 downto 0);
        Addr_RAM_Phys: UNSIGNED(WIDTH_ADDR-1 downto 0);
        Data4RAM: UNSIGNED(WIDTH_WORD-1 downto 0);
        Data2RAM: UNSIGNED(WIDTH_WORD-1 downto 0);
        -------------- signals connecting with Memory -------------
        Hist_RAM: in STD_LOGIC_VECTOR(WIDTH_word-1 downto 0);
    end CELL;

architecture STRUCT of CELL is
    -- signals coming from the global controller needs to be registered
    signal Suffix_Reg_en_L1: STD_LOGIC;
    signal Prefix_en_L1: STD_LOGIC;
    signal Suffix_tran_en_L1: STD_LOGIC;
    signal Tran_ing_L1: STD_LOGIC;
    signal Comp_en_L1: STD_LOGIC;
    signal Cnt_Rd_L1: INT_RD;
    signal Cnt_Comp_L1: INT_PIXEL_ALL;
    signal Wr_RAM_Req_L1: STD_LOGIC;

    -- signals between MMU and local controller ------------------
    signal Rd_RAM_en_Vec: UNSIGNED(NUM_PIXEL-1 downto 0);
    signal Pixel_sel_vec: UNSIGNED(NUM_PIXEL-1 downto 0);
    signal Rd_RAM_Req_Vec: UNSIGNED(NUM_PIXEL-1 downto 0);
    signal Rd_Lib_en_Vec: UNSIGNED(NUM_PIXEL-1 downto 0);
    signal Addr_RAM_Vec: ADDR_VECT(NUM_PIXEL-1 downto 0);
    signal Data_RAM_Vec: WORD_VECT(NUM_PIXEL-1 downto 0);
    signal Overlap_Vec: OVERLAP_VECT(NUM_PIXEL-1 downto 0);

    -- signals with Memory -------------------------------------
    signal History_RAM: STD_LOGIC_VECTOR(WIDTH_word-1 downto 0);
    signal s: STD_LOGIC_VECTOR(9 DOWNTO 0);
begin
    process(Clk)
    begin
        if rising_edge(Clk) then
            Suffix_Reg_en_L1<=Suffix_Reg_en;
            Prefix_en_L1<=Prefix_en;
            Suffix_tran_en_L1<=Suffix_tran_en;
            Tran_ing_L1<=Tran_ing;
            Comp_en_L1<=Comp_en;
            Cnt_Rd_L1<=Cnt_Rd;
            Cnt_Comp_L1<=Cnt_Comp;
            Wr_RAM_Req_L1<=Wr_RAM_Req;
        end if;
    end process;
end STRUCT;
```
end process;

---------- instantiation of pixel array ----------/
ARRAY_INST: entity PIXELARRAY port map
  (
    Clk=>Clk, Rst=>Rst,
    Data_in=>Data_in,
    Suffix_Reg_en=>Suffix_Reg_en_L1,
    Prefix_en=>Prefix_en_L1,
    Suffix_tran_en=>Suffix_tran_en_L1,
    Tran.in=>Tran.in.L1,
    Comp.en=>Comp.en.L1,
    Cnt.Rd=>Cnt.Rd.L1,
    Cnt.Cmp=>Cnt.Cmp.L1,
    Overlap.Vec=>Overlap.Vec,
    Rd.RGB_en=>Rd.RGB_en,L1,
    Rd.RGB_enc=>Rd.RGB_enc,L1,
    Rd.RGB Req.Vec=>Rd.RGB Req.Vec,L1,
    Addr.RGB.Vec=>Addr.RGB.Vec,L1,
    Overlap2FIFO=>Overlap2FIFO,L1
    );

---------- instantiation of memory Scheduler ----------/
SCHEDULER_INST: entity SCHEDULER port map
  (
    Clk=>Clk, Rst=>Rst,
    -- ports connecting with data source
    Rd_lib_req=>Rd_lib_req,
    Rd_lib_ack=>Rd_lib_ack,
    -- ports connecting with global FIFO
    Addr.Virt.Vec=>Addr.RGB.Vec,
    Rd.RGB Req.Vec=>Rd.RGB Req.Vec,L1,
    Overlap.Vec=>Overlap.Vec,L1,
    Data.RGB.Vec=>Data.RGB.Vec,L1,
    Rd.RGB en.Vec.L1=>Rd.RGB en.Vec,L1,
    Rd_lib.en.vc=>Rd_lib.en.vc,
    -- ports connecting with global controller
    Wr.RGB Req=>Wr.RGB Req.L1,
    -- ports connecting with Pixels
    Data2RAM=>Data2RAM,
    Addr_phys=>Addr.RGB.Phys,
    Wr2RAM.L1=>Wr2RAM.L1,
    Wr2FIFO.L1=>Wr2FIFO.L1
 );

---------- instantiation of RAM ----------/
RAM_INST: entity RAM port map
  (
    address=>std_logic_vector(Addr.RGB.Phys),
    clock=>Clk,
    data=>data,
    wren=>Wr.RGB.en,
    q=>q
  );

---------- signal wrapping ----------
data=>std_logic_vector(Data2RAM),
Data4RAM=>unsigned(q);

end STRUCT;

Listing D.5: FIFO Management
which somehow averages the number that every pixel has been selected. If a writing process starts,
it will keep doing till whole writing processes have been processed..

Called by: SYSTEM.vhd

Revision history: 1.0

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;

entity FIFO_MANAGE is
port(
  Clk,Rst: in STD_LOGIC;
  Rd_Lib_Req: in STD_LOGIC;
  Rd_Lib_ack: out STD_LOGIC;
  Rd_FIFO_ack_vec: in UNSIGNED(NUM_CELL-1 downto 0);
  Wr2FIFO_Vec: in UNSIGNED(NUM_CELL-1 downto 0);
  Overlap_Vec: in OVERLAP_VECT(NUM_CELL-1 downto 0);
  Rd_FIFO_req_vec: out UNSIGNED(NUM_CELL-1 downto 0);
  Wr2FIFO_en: out STD_LOGIC;
  Overlap_o: out OVERLAP);
end FIFO_MANAGE;

architecture RTL of FIFO_MANAGE is

type MULTI_STATE is (IDLE,SWITCH,RD_FIFO);
signal ss,ss_next: MULTI_STATE;
signal Cnt_Cell: INT_CELL; -- the previous selected number
signal Cnt_Cell_flag: STD_LOGIC;
signal Rd_FIFO_ack: STD_LOGIC;

begin
  SS_PROC: process(Rst,Clk)
  begin
    if Rst='1' then
      ss<=IDLE;
    elsif rising_edge(Clk) then
      ss<=ss_next;
    end if;
  end process SS_PROC;

  SS_NEXT_PROC: process(ss,Rd_Lib_req,Rd_FIFO_ack)
  begin
    ss_next<=ss;
    case ss is
      when IDLE=>
        if (Rd_Lib_req='1') then -- have write request and not all have been processed
          ss_next<=RD_FIFO;
        end if;
      when SWITCH=>
        if Rd_FIFO_ack='0' then
          ss_next<=RD_FIFO;
        end if;
        if (Rd_Lib Req='0') then
          ss_next<=IDLE;
        end if;
      when RD_FIFO=>
        if Rd_FIFO_ack='1' then
          ss_next<=SWITCH;
        end if;
      when others=>
        null;
    end case;
    end process SS_NEXT_PROC;

  PROC: process(Rst, Clk)
  begin
    if Rst='1' then
      -- reset output signals
      Rd_Lib_ack<=0';
      Rd_FIFO_req_vec<=(others=>'0');
      Wr2FIFO_en<=0';
  end process;
end process;

Overlap_o<=(others=>'0'); -- reset internal signal for computation
Rd_FIFO_ack<=0;
Cnt_Cell:=0;
Cnt_Cell_flag<='0';
elseif rising_edge(Clk) then
  case ss is
  when IDLE=>
    Cnt_Cell<=0;
    when SWITCH=>
      Rd_FIFO_ack<=Rd_FIFO.ack(vec(Cnt_Cell));
      if Cnt_Cell_flag='0' then
        Cnt_Cell_flag<='1';
        if Cnt_Cell=INT_CELL'HIGH then
          Cnt_Cell:=Cnt_Cell+1;
        else
          Rd_Lib.ack<='1';
        end if;
        if Cnt_Cell<INT_CELL'HIGH then
          Cnt_Cell<=Cnt_Cell+1;
        end if;
      end if;
      when RD_FIFO=>
        Cnt_Cell_flag<='0';
        Rd_FIFO_req_vec<=(others=>'0');
        Rd_FIFO_req_vec(Cnt_Cell)<='1';
        Wr2FIFO_en<= Wr2FIFO.Vec(Cnt_Cell);
        Rd_FIFO.ack<='1';
        when others=>null;
      end case;
      end if;
end if;
end process PROC;

Listing D.6: Pixel Array

entity PIXELARRAY is
port(
  Clk, Rst: in STD_LOGIC;
  ------------ signals from the sources -------------
  Data_in: in DATA_ARRAY(NUM_PIXEL-1 downto 0);
  ------------ signals with global controller -----------
  Suffix.Reg.en: in STD_LOGIC;
  Prefix.en: in STD_LOGIC;
  Suffix.tran.en: in STD_LOGIC;
  Tran.ing: in STD_LOGIC;
  Comp.en: in STD_LOGIC;
  Cnt.Rd: in INT.RD;
  Cnt.Comp: in INT_PIXEL.ALL;
  ------------ signals with MMU ------------------------
  Rd_RAM_en.Vec: in UNSIGNED(NUM_PIXEL-1 downto 0);
  Rd_Lib.en.vec: in UNSIGNED(NUM_PIXEL-1 downto 0);
  Rd_RAM.Reg.Vec: out UNSIGNED(NUM_PIXEL-1 downto 0);
  Addr.RAM.Vec: out ADDR.VECT(NUM_PIXEL-1 downto 0);
  ------------ signals with Memory ---------------------
  Data4RAM: in UNSIGNED(WIDTH.WORD-1 downto 0);
  Data_RAM.Vec: out WORD.VECT(NUM_PIXEL-1 downto 0);
  Overlap.Vec: out OVERLAP.VECT(NUM_PIXEL-1 downto 0);
  ------------ signals with other pixel cluster ----------
  Suffix.i: in SEQ.DNA(LENG_FIX-1 downto 0);
  Suffix.o: out SEQ.DNA(LENG_FIX-1 downto 0)
);
end PIXELARRAY;

architecture STRUCT of PIXELARRAY is

component PIXEL port:
   Clk, Rst: in STD_LOGIC;
   Data.in: in BASE;
   Suffix.Reg.en: in STD_LOGIC;
   Prefix.en: in STD_LOGIC;
   Suffix.trans.en: in STD_LOGIC;
   Tran.in: in STD_LOGIC;
   Comp.en: in STD_LOGIC;
   Cnt.Rd: in INT_RD;
   Cnt.Comp: in INT_PIXEL.ALL;
   Overlap.o: out OVERLAP;
   Suffix.Reg.en.L1: STD_LOGIC;
   Prefix.en.L1: STD_LOGIC;
   Suffix.trans.en.L1: STD_LOGIC;
   Tran.in.L1: STD_LOGIC;
   Comp.en.L1: STD_LOGIC;
   Cnt.Rd.L1: INT_RD;
   Cnt.Comp.L1: INT_PIXEL.ALL;

begin
   process(Clk)
   begin
      if rising_edge(Clk) then
         Prefix.en.L1<=Prefix.en;
         Suffix.trans.en.L1<=Suffix.trans.en;
         Tran.in.L1<=Tran.in;
         Comp.en.L1<=Comp.en;
         Cnt.Rd.L1<=Cnt.Rd;
         Cnt.Comp.L1<=Cnt.Comp;
      end if;
   end process;

   type suffixarray is array (NUM_PIXEL-2 downto 0) of SEQ_DNA(LENG_FIX-1 downto 0);
   signal suffix: suffixarray;
   signal Suffix.Reg_en_L1: STD_LOGIC;
   signal Prefix_en_L1: STD_LOGIC;
   signal Suffix_trans_en_L1: STD_LOGIC;
   signal Tran_ing_L1: STD_LOGIC;
   signal Comp_en_L1: STD_LOGIC;
   signal Cnt_Rd_L1: INT_RD;
   signal Cnt_Comp_L1: INT_PIXEL.ALL;

begin
   process(Clk)
   begin
      if rising_edge(Clk) then
         Prefix.en.L1<=Prefix.en;
         Suffix.trans.en.L1<=Suffix.trans.en;
         Tran.in.L1<=Tran.in;
         Comp.en.L1<=Comp.en;
         Cnt.Rd.L1<=Cnt.Rd;
         Cnt.Comp.L1<=Cnt.Comp;
      end if;
   end process;

   begin
      if rising_edge(Clk) then
         Prefix.en.L1<=Prefix.en;
         Suffix.trans.en.L1<=Suffix.trans.en;
         Tran.in.L1<=Tran.in;
         Comp.en.L1<=Comp.en;
         Cnt.Rd.L1<=Cnt.Rd;
         Cnt.Comp.L1<=Cnt.Comp;
      end if;
   end process;

   begin
      if rising_edge(Clk) then
         Prefix.en.L1<=Prefix.en;
         Suffix.trans.en.L1<=Suffix.trans.en;
         Tran.in.L1<=Tran.in;
         Comp.en.L1<=Comp.en;
         Cnt.Rd.L1<=Cnt.Rd;
         Cnt.Comp.L1<=Cnt.Comp;
      end if;
   end process;

end component;

begin
   process(Clk)
   begin
      if rising_edge(Clk) then
         Prefix.en.L1<=Prefix.en;
         Suffix.trans.en.L1<=Suffix.trans.en;
         Tran.in.L1<=Tran.in;
         Comp.en.L1<=Comp.en;
         Cnt.Rd.L1<=Cnt.Rd;
         Cnt.Comp.L1<=Cnt.Comp;
      end if;
   end process;

end PIXELARRAY;
PIXEL_TOP: PIXEL port map(
  Clk=>Clk, Rst=>Rst,
  ------------ signals with source circuits -----------
  Data_in=Data_in(NUM_PIXEL-1),
  ---------------- signals with global controller --------
  Suffix_Reg_en=Suffix_Reg_en_L1,
  Prefix_en=Prefix_en_L1,
  Suffix_tran_en=Suffix_tran_en_L1,
  Tran_in=Tran_in_L1,
  Comp_en=Comp_en_L1,
  Cnt_Rd=Cnt_Rd_L1,
  Cnt_Comp=Cnt_Comp_L1,
  Overlap=>Overlap_VEC(NUM_PIXEL-1),
  -------------------- signals with MMU ----------------
  Rd_RAM_en=Rd_RAM_en_VEC(NUM_PIXEL-1),
  Nd_RAM_Req=Rd_RAM_Req_VEC(NUM_PIXEL-1),
  Addr_RAM=Addr_RAM_VEC(NUM_PIXEL-1),
  ---------------- signals with Memory ----------------
  Data_RAM_Lo=Data_RAM_Lo,
  Data_RAM_hi=Data_RAM_Hi_VEC(NUM_PIXEL-1),
  ---------------- signals with other pixels -----------
  Suffix_i=SuffixVEC(NUM_PIXEL-2),
  Suffix_o=>Suffix_o);
)

PIXEL_BOTTOM: PIXEL port map(
  Clk=>Clk, Rst=>Rst,
  ------------ signals with source circuits -----------
  Data_in=Data_in(0),
  ---------------- signals with global controller --------
  Suffix_Reg_en=Suffix_Reg_en_L1,
  Prefix_en=Prefix_en_L1,
  Suffix_tran_en=Suffix_tran_en_L1,
  Tran_in=Tran_in_L1,
  Comp_en=Comp_en_L1,
  Cnt_Rd=Cnt_Rd_L1,
  Cnt_Comp=Cnt_Comp_L1,
  Overlap=>Overlap_VEC(0),
  -------------------- signals with MMU ----------------
  Rd_RAM_en=Rd_RAM_en_VEC(0),
  Nd_RAM_Req=Rd_RAM_Req_VEC(0),
  Addr_RAM=Addr_RAM_VEC(0),
  ---------------- signals with Memory ----------------
  Data_RAM_Lo=Data_RAM_Lo,
  Data_RAM_hi=Data_RAM_Hi_VEC(0),
  ---------------- signals with other pixels -----------
  Suffix_i=Suffix_i,
  Suffix_o=>Suffix_o);
end STRUCT;

Listing D.7: Scheduler

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;
extend entity SCHEDULER is
port(
  Clk, Rst: in STD_LOGIC;
  ---------------- ports connecting with Library fetching management unit --------
  Rd_Lib_req: in STD_LOGIC;
  Rd_Lib_ack: out STD_LOGIC;
  ---------------- ports connecting to the global FIFO
  overlap2FIFO: out OVERLAP;
);
architecture RTL of SCHEDULER is

type MULTI_STATE is (IDLE, SEL, RD_1, RD_2, WR, RD_LIB);
subtype INT_LIB is INTEGER range 0 to 15;

signal ss, ss_next: MULTI_STATE;
signal Pntr_Rd: INT PIXEL; -- the previous selected number
signal Pntr_Wr: INT PIXEL;
signal Cnt_Wr: INT RD; -- word address in writing
signal Pntr.dupl: UNSIGNED(1 downto 0); -- signal used to avoid the same read request
signal Rd_RAM_en_Vec: UNSIGNED(NUM_PIXEL-1 downto 0);
signal Pixel_sel_Vec: UNSIGNED(NUM_PIXEL-1 downto 0);
signal Pntr.Rd.Lib: INT PIXEL;
signal Cnt_Rd.Lib: INT_LIB;
signal Wr_RAM.req.label:STD_LOGIC;
signal Sel.ready: STD_LOGIC;
signal Wr2RAM: STD_LOGIC;
signal Wr2FIFO: STD_LOGIC;
signal Addr.Virt.o: UNSIGNED(WIDTH_ADDR-1 downto 0);

begin
SS_PROC: process(Rst, Clk)
begin
if Rst='1' then
ss<=IDLE;
elsif rising_edge(Clk) then
ss<=ss_next;
end if;
end process SS_PROC;

SS_NEXT_PROC: process(ss, Wr_RAM Req.label, Sel.ready, Rd_RAM Req.Vec, Pntr.dupl, Rd.Lib.req)
begin
ss.next<=>ss;
begin
    case ss is
        when IDLE=>
            if (Wr_RAM Req.label='1') then -- have write request and not all have been processed
                ss.next<=WR;
            elsif nonzero(Rd_RAM Req.Vec) then -- have read request
                ss.next<=SEL;
            end if;
        when RD_1=>
            if Rd.Lib.req='1' then
                ss.next<=RD_LIB;
            end if;
        when RD_2=>
            if Rd.Lib.req='0' then
                ss.next<=IDLE;
            end if;
        when SEL=>
            if Sel.ready='1' then
                if Pntr.dupl='01' then
                    ss.next<=RD_1;
                else
                    ss.next<=IDLE;
                end if;
            end if;
        when RD.Lib=>
            if Rd.Lib.req='1' then
                ss.next<=RD_LIB;
            end if;
        when others=>
            null;
    end case;
end process SS_NEXT_PROC;

DATA_PROC: process(Clk, Rst)
begin
variable addr.block: INTEGER;
end process DATA_PROC;

end SCHEDULER;
variable Pntr.Rd.next: INT_PIXEL;
begin
  if Rst='1' then
    -- reset the internal signal
    Pntr.dupl<="00';
    Wr_RAM Req.label<='0';
    Pntr.Rd<='0';
    Pntr.Wr<='0';
    Sel.ready<='0';
    -- signal for lib reading
    Pntr.Rd.Lib<='0';
    Cnt.Rd.Lib<='0';
  reset the output ports
  Rd.Ram.en.Vec<=others=>'0';
  Pixel.sel.Vec<=others=>'0';
  Rd.Lib.en.vec<=others=>'0';
  Addr.phys<=others=>'0';
  -- signal for lib reading
  Wr2RAM<='0';
  Wr2FIFO<='0';
  RdLib.ack.s<='0';
  elsif rising_edge(Clk) then
    case ss
      is
      when IDLE =>
        Rd.Ram.en.Vec<=(others=>'0');
        RdLib.en.Vec<=(others=>'0');
        Pixel.sel.Vec<=(others=>'0');
        Wr2RAM<='0';
        Wr2FIFO<='0';
        Sel.ready<='0';
        if ss_next=SEL then
          Pntr.Rd_next:=index_Sel(Rd_RAM_Req_Vec,Pntr.Rd);
          Pntr.Rd<= Pntr.Rd_next; -- choose the next pixel in a circular way
          if Pntr.Rd=Pntr.Rd_next then
            Pntr.dupl<= Pntr.dupl+1;
            else
            Pntr.dupl<="01' ;
          end
        end
      when SEL =>
        Sel.ready<='1';
        Pixel.sel.Vec(Pntr.Rd)<= '1'; -- utilise the write enable signal to get the Virtual Address
      when RD_1 =>
        addr.block:=Pntr.Rd*(LENG_READ/LENG_FIX+1);
        Addr.phys<=to_unsigned(addr.block+to_integer(Addr_Virt_o),WIDTH_ADDR); -- Calculate the physical address
        Rd.Ram.en.Vec(Pntr.Rd)<= '1'; -- enable reading process
      when RD_2 =>
        Addr.phys<=to_unsigned(addr.block+to_integer(Addr_Virt_o)+1,WIDTH_ADDR); -- Calculate new physical address
      when WR =>
        addr.block:=Pntr.Wr*(LENG_READ/LENG_FIX+1);
        Pixel.sel.Vec(Pntr.Wr)<= '1'; -- ask the mux to choose the data out from the array
        Addr.phys<=to_unsigned(addr.block+Cnt.Wr,WIDTH_ADDR);
        Wr2RAM<='1'; -- write enable signal for the RAM
        if Pntr.Wr=INT_PIXEL'HIGH then
          Pntr.Wr<='0';
          Wr_RAM Req.label<='0';
          Cnt.Wr<=Cnt.Wr+1; -- when all writing processes are finished, add one block address
        else
          Pntr.Wr<=Pntr.Wr+1; -- select pixel one by one in writing
        end
      when RD_LIB =>
        Pixel.sel.Vec<=others=>'0';
        Pixel.sel.Vec(Pntr.Rd.Lib)<= '1';
        Rd.Lib.en.Vec<=others=>'0';
        RdLib.en.Vec<=Pntr.Rd.Lib*1';
        Wr2FIFO<='1';
        if Cnt.Rd.Lib(15 downto 0) and RdLib.ack.s='0' then -- read all the seeds in the seed library
          Cnt.Rd.Lib<=Cnt.Rd.Lib+1; -- when all writing processes are finished, add one word address
        else
          RdLib.ack.s<= '1';
          Wr2FIFO<='0';
        end
      when RD_LIB <= INT_PIXEL'HIGH then
        Pntr.Rd.Lib<=Pntr.Rd.Lib+1; -- select pixel one by one in writing
      end

end if;
when others=>
null;
end case;
if Wr_RAM Req='1' then
-- setup the RAM write request
end if;
----------------- one clock delay ----------------/
Rd_RAM_en_Vec_L1<=Rd_Ram_en_Vec;
Wr2RAM_L1<=Wr2RAM;
Wr2FIFO_L1<=Wr2FIFO;
Wr2FIFO_L2<=Wr2FIFO_L1;
Rd.Lib.ack<=Rd.Lib.ack_s;
end if; -- end of clock
----------------- zero clock delay ----------------/

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;
use WORK.all;
entity PIXEL is
port(
    Clk, Rst: in STD_LOGIC;
    Data_in: in BASE;
    Suffix_Reg_en: in STD_LOGIC;
    Prefix_en: in STD_LOGIC;
    Suffix_tran_en: in STD_LOGIC;
    Tran_ing: in STD_LOGIC;
    Comp_en: in STD_LOGIC;
    Cnt Rd: in INT_RD;
    Cnt.Comp: in INT PIXEL_ALL;
    Overlap_o: out OVERLAP;
    Rd_RAM_en: in STD_LOGIC;
    Rd.Lib.en: in STD_LOGIC;
    Rd_RAM_Req: out STD_LOGIC;
    Addr_RAM: out UNSIGNED(WIDTH_ADDR-1 downto 0);
    Data_RAM_i: in UNSIGNED(WIDTH_WORD-1 downto 0);
    Data_RAM_o: out UNSIGNED(WIDTH_WORD-1 downto 0);
    Suffix_i: in SEQ DNA(LENG_FIX-1 downto 0);
    Suffix_o: out SEQ DNA(LENG_FIX-1 downto 0)
);
end PIXEL;
architecture STRUCT of PIXEL is
component DATAPATH port(
  Clk, Rst: in STD_LOGIC;
  -- ports connecting with peripheral circuits
  Data_in: in BASE;               -- signal from sensor sources
  Data_RAM_i: in UNSIGNED(WIDTH_WORD-1 downto 0);
  Data_RAM_o: out UNSIGNED(WIDTH_WORD-1 downto 0);
  -- ports connecting with other pixels
  Suffix_i: in SEQ_DNA(LENG_FIX-1 downto 0);
  Suffix_o: out SEQ_DNA(LENG_FIX-1 downto 0);
  -- ports connecting with global controller
  Prefix.en: in STD_LOGIC;
  Suffix_tran.en: in STD_LOGIC;
  Suffix.Reg.en: in STD_LOGIC;
  Comp.en: in STD_LOGIC;
  -- ports connecting with local controller
  DynP.en: in STD_LOGIC;
  Matching: out STD_LOGIC;
  Suffix.chk.en: in STD_LOGIC;
  Infix.x2.en: in STD_LOGIC;
  Infix.Num: in STD_LOGIC;
  Infix.en: in STD_LOGIC;
  Infix.Reg: in STD_LOGIC;
  Infix.shift: in INT SHIFT; Infix;
  DynP.done: out STD_LOGIC;
  -- ports connecting with seeds library
  Shift.o: out INT SHIFT;
  Score.o: out INT EdDi;
  Bases.i: in VECT INT EdDi(REDDUN*2 downto 0);
  Bases.o: out VECT INT EdDi(REDDUN*2 downto 0)
); end component;

begin
  process(Clk)
    begin
      if rising_edge(Clk) then
        suffix.Reg.en_L1<=Suffix.Reg.en;
        prefix.en.L1<=Prefix.en;
        suffix_tran.en.L1<=Suffix_tran.en;
        transing.L1<=Transing;
        comp.en.L1<=Comp.en;
        Cnt.Rd.L1<=Cnt.Rd;
        Cnt.Comp.L1<=Cnt.Comp;
      end if;
  end process;
end component;
Listing D.9: Multiplexer

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;

entity MUX_CELL is
  port(
    Clk: in STD_LOGIC;
    -- signals from the array of pixels
    Overlap_Vec: in OVERLAP_VECT(NUM_PIXEL-1 downto 0);
    Addr_Virt_Vec: in ADDR_VECT(NUM_PIXEL-1 downto 0);
    Data_RAM_Vec: in WORD_VECT(NUM_PIXEL-1 downto 0);
    Pixel_sel_Vec: in UNSIGNED(NUM_PIXEL-1 downto 0);
    Addr_Virt_o: out UNSIGNED(WIDTH_ADDR-1 downto 0);
    Data_RAM_o: out UNSIGNED(WIDTH_ADDR-1 downto 0);
  );
end STRUCT;

-- FILE name: MUX.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 14/06/2012
-- version: 1.0
-- Abstract: This file describes a combinational MUX. The necessity of this block is that there is no databus in FPGA, to
-- use a global RAM, we need to choose the datapath between the pixels and RAM.
-- Called by: PIXELARRAY.vhd
-- Revision history: N/A
```
architecture rtl of MUX_CELL is
begin
  sel: process(Clk)
  begin
    if rising_edge(Clk) then
      Addr_Virt_o<=Addr_Virt_Vec(0);
      Data_RAM_o<=Data_RAM_Vec(0);
      Overlap_o<=Overlap_Vec(0);
      for i in NUM_PIXEL-1 downto 0 loop
        if Pixel_sel_Vec(i)='1' then
          Addr_Virt_o<=Addr_Virt_Vec(i);
          Data_RAM_o<=Data_RAM_Vec(i);
          end if;
      end loop;
      LIB_READ: for i in NUM_PIXEL-1 downto 0 loop
        if Pixel_sel_Vec(i)='1' then
          Overlap_o<=Overlap_Vec(i);
          end if;
      end loop LIB_READ;
    end if;
  end process sel;
end rtl;

Listing D.10: Datapath
architecture RTL of DATAPATH is
component ALU port(
    Clk: in STD_LOGIC;
    Comp_en: in STD_LOGIC;
    Word_V: in SEQ_DNA(LENG_FIX-1 downto 0);
    Word_H: in SEQ_DNA(LENG_FIX-1 downto 0);
    Bases_i: in VECT_INT_EdDi(REDUN*2 downto 0);
    Comp_done: out STD_LOGIC;
    Bases_o: out VECT_INT_EdDi(REDUN*2 downto 0);
    Score_o: out INT_EdDi;
    Shift_o: out INT SHIFT
); end component;
signal Infix_shift_Cnt: INT_SHIFT_INFIX;
signal score: INT_EdDi;
signal prefix: SEQ_DNA(LENG_FIX-1 downto 0);
signal suffix_ext: SEQ_DNA(LENG_FIX-1 downto 0);
signal suffix_chk: SEQ_DNA(LENG_FIX-1 downto 0);
signal Suffix_Reg_x2: SEQ_DNA(LENG_FIX*2-1 downto 0);
signal infix: SEQ_DNA(LENG_FIX*2-1 downto 0);
begin
    ALU_0: ALU port map(
        Clk=>Clk,
        Comp_en=>DynP_en,
        Word_H =>suffix_chk, Word_V=>infix,
        Bases_i=>bases_i,
        Comp_done=>DynP_done,
        Bases_o=>Bases_o,
        Score_o=>Score_o,
        Shift_o=>Shift_o
    );
data_proc: process(Clk,Rst)

    variable infix_var: SEQ_DNA(LENG_FIX-1 downto 0);
    begin
        if Rst='1' then -- clear all the register
            prefix<=(others=>'00');
            infix<=(others=>'00');
            infix_x2<=(others=>'00');
            suffix_ext<=(others=>'00');
            suffix_chk<=(others=>'00');
            Suffix_Reg_x2<=(others=>'00');
        elsif rising_edge(Clk) then
            if Prefix_en='1' then -- shift in prefix
                prefix(prefix'HIGH downto 1)<=prefix(prefix'HIGH-1 downto 0);
                prefix(0)<=Data_in;
            end if;
            if Suffix_Reg_en='1' then -- register for RAM writing
                Suffix_Reg_x2(Suffix_Reg_x2'HIGH downto 1)<=Suffix_Reg_x2(Suffix_Reg_x2'HIGH-1 downto 0);
                Suffix_Reg_x2(0)<=Data_in;
                suffix_ext(suffix_ext'HIGH downto 1)<=suffix_ext(suffix_ext'HIGH-1 downto 0);
            end if;
            if Suffix_tran_en='1' then -- transmit suffix among pixels
                suffix_ext<=suffix_i;
            end if;
            if Suffix_chk_en='1' then -- copy the suffix from the sequence which will be checked later
                suffix_chk<=suffix_ext;
            end if;
        end if;
    end process;
begin
    if Infix_x2_en='1' then -- read the first word
        infix_var:=unsigned_to_seq(Data_RAM_i);
        infix_x2(2*LENG_FIX-1 downto 0)<=infix_var; -- small-index word locates higher
    else -- read the second word
        infix_var:=unsigned_to_seq(Data_RAM_i);
        infix_x2(2*LENG_FIX-1 downto 0)<=infix_var;
    end if;
end
end
end process;
end if;
end if;
---------------------------------------------------------
if Infix_en='1' then
if Infix_from_Reg='0' then
  infix<=infix_x2(LENG_FIX+Infix_shift-1
downto
Infix_shift);
else
  infix<=suffix_Reg_x2(LENG_FIX+Infix_shift-1
downto
Infix_shift);
end if;
end if;
---------------------------------------------------------
if Comp_en='1'
  and
  suffix_ext=prefix then
  Matching<='1';
else
  Matching<='0';
end if;
end if; -- end of clock
end process DATA_PROC;
---------------------------------------------------------
Data_RAM_o<=seq_to_unsigned(Suffix_Reg_x2(LENG_FIX-1
downto
0));
Suffix_o<=suffix_ext;
end RTL;

Listing D.11: Local Controller

-- FILE name: LOCAL_CTRL.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 13/06/2012
-- version: 5.0
-- Abstract: This file is the local control part in every pixel, which comprises two FSMs-- library organization and
RAM reading.
-- As the names imply, the Library FSM governs the seeds library that stores all the potential overlaps, and
-- used to received the data from RAM.
-- The overlap seeds are found by monitoring the comparison results from the Datapath. A valid "matching"
-- exact match between two sub-strings. While the checking process utilises the inexact comparison
-- When the "Chk_done" is active, the compared scores will be examined with a threshold value, once the
-- storing enable signal "Matching" is set to the Library.
-- System is optimised for operating frequency
-- The RAM reading request is sented from the Lib FAM as well, which needs RAM-stored sequences to check the
-- In this version, the checking process is executed at the same time with the seeds searching process.
-- Some abbreviations: Comp=compare, ext=external, int=internal, Req=request, Lib=library
-- Sour=source, Tran=transmission
-- Called by: Pixel.vhd
-- Revision history: 3.5 4.0
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.all;
use WORK.PARAMETERS.all;
use WORK.FUNCTIONS.all;
entity LOCAL_CTRL is
  port(
    Clk, Rst: in STD_LOGIC;
    Suffix_tran_en:
in  STD_LOGIC;
    Cnt_Rd: in INT_RD;
    Cnt_Comp: in INT_PIXEL_ALL;
    Tran_ing: in STD_LOGIC;
    Overlap.o: out OVERLAP;
    Rd_RAM_en: in STD_LOGIC;
    Rd_Lib_en: in STD_LOGIC;
    Rd_RAM Req: out STD_LOGIC;
    Addr_RAM: out UNSIGNED(WIDTH_ADDR-1
downto
0);
    Shift4DP: in INT_SHIFl;
    Score.i: in INT_EdDi;
    Bases4DP: in VECT_INT_EdDi(REDUN*2
downto
0);
    Bases2DP: out VECT_INT.EdDi(REDUN*2
downto
0);
-- enable signal
Matching: in STD_LOGIC; -- exact comparison indication
DynP_done: in STD_LOGIC;
DynP_en: out STD_LOGIC;
Infix_en: out STD_LOGIC;
Infix_x2_en: out STD_LOGIC;
Infix_num: out STD_LOGIC;
Infix_from_reg: out STD_LOGIC;
Infix_shift: out INT_SHIFT_INFIX; -- the wanted word stays in the middle of
-- the two-concatenated long word with certain shift
Suffix_chk_en: out STD_LOGIC -- store the data used to compared with the infix during transmission
end LOCAL_CTRL;
architecture RTL of LOCAL_CTRL is
  type MULTI_STATE_Lib is (IDLE_Lib, FETCH_RAM, FETCH_REG, CALCUL, CHECK, WAIT, CHECK_WAIT, LIB_WAIT);
  type MULTI_STATE_RAM is (IDLE_RAM, ADDR_RAM, RD_RAM_1, RD_RAM_2);

  -- signals used for library control
  signal ss_Lib, ss_next_Lib: MULTI_STATE_Lib;
  -- to the library
  signal Rd_seed: STD_LOGIC;
  signal Chk_done: STD_LOGIC;
  signal Chk_result: STD_LOGIC;
  signal position2Lib: INT_RD;
  signal rotate2Lib: INT_PIXEL_ALL;
  signal score2Lib: INT_ERROR;
  signal bases2Lib: VECT_INT_EDDi(REDUN*2 downto 0);
  -- from the library
  signal Lib_idle: STD_LOGIC;
  signal seed4Lib: OVERLAP;
  signal Chk_wt: INT_WAIT; -- cycles waited since last check
  signal position4Lib: INT_RD;
  signal rotate4Lib: INT_PIXEL_ALL;
  signal score4Lib: INT_ERROR;
  signal bases4Lib: VECT_INT_EDDi(REDUN*2 downto 0);

  -- signals used for the RAM data reading
  signal ss_RAM, ss_next_RAM: MULTI_STATE_RAM;
  signal read_ack: STD_LOGIC; -- has read the infix from the RAM
  signal Addr_RAM_Reg: UNSIGNED(WIDTH_ADDR-1 downto 0); -- the address for RAM reading
  -- flag signal used locally for checking and calculation
  signal Calcu_Cnt: STD_LOGIC; -- allow state CALC to last two clock cycles
  signal Tran_ing_L1: STD_LOGIC; -- used to generate one clock cycle done signal
  signal suffix_chk.lock: STD_LOGIC;
  signal infix_reg: STD_LOGIC; -- same as Infix_frm.Reg
  signal suffix Tran_en_L1: STD_LOGIC;
  signal Rd_done: STD_LOGIC;
  signal cycle done: STD_LOGIC;
  signal next cycle: STD_LOGIC; -- signal annotating whether checking takes place at next cycle
  signal ready.infix: STD_LOGIC;
  signal ready.suffix: STD_LOGIC;
  signal ready.RAM: STD_LOGIC;
  signal ready.Reg: STD_LOGIC;
  signal distance: INT_RD; -- internal signal, the distance of the infix from the head of the sequence

begin
  Library_0: entity Lib_0 entity Lib_0 port map
  (Clk=>Clk, Rst=>Rst,
   -- signals from global controller
   Cnt_Rd=>Cnt_Rd, Cnt_Comp=>Cnt_Comp,
   Matchng=>Matching,
   -- signal from intra-pixel
   Cycle done=>cycle done, Rd_seed=>Rd_seed, Chk_result=>Chk_result,
   -- data to store
   Position_i=>position2Lib, Rotate_i=>rotate2Lib,
   Score_in=>score2Lib, Bases_in=>Bases2Lib,
   -- signal to local pixel
   Lib idle=>Lib idle,
   Seed_o=>seed4Lib, Chk_Wt_o=>Chk_wt);

  begin
    if Rst='1' then
      ss_Lib<=IDLE_Lib;
    end if;
end Library_0;
ss_RAM<=IDLE_RAM;
elsif rising_edge(Clk) then
  ss_Lib<=ss.next_Lib;
  ss_RAM<=ss.next_RAM;
end if;
end process SS_PROC;

------------ calculate the next state for reading the data from RAM --------------
SS_NEXT_RAM_PROC: process(ss_RAM,Rd_RAM_en)
begin
  ss.next_RAM<=ss_RAM;
  case ss_RAM
  is
    when IDLE_RAM =>
      Infix_x2_en<='0'; -- combinaional enable output signal -
      if Rd_RAM_en='1' then
        ss.next_RAM<=Rd_RAM_1; -- reading phase one
      end if;
    when Rd_RAM_1 =>
      Infix_x2_en<='1';
      if Rd_RAM_en='1' then
        ss.next_RAM<=RD_RAM_2; -- reading phase two
      end if;
    when Rd_RAM_2 =>
      Infix_x2_en<='1';
      ss.next_RAM<=IDLE_RAM; -- back to idle state
    when others =>
      Infix_x2_en<='0';
  end case;
end process SS_NEXT_RAM_PROC;

----------------- RAM data reading process ----------------------
DATA_RAM_PROC: process(Clk,Rst)
begin
  if Rst='1' then
    Infix_Num<='0'; -- reading order signal
    read_ack<='0'; -- reading finish signal
  elsif rising_edge(Clk) then
    case ss_RAM
    is
      when IDLE_RAM =>
        Infix.Num<='0'; -- clear the internal signal or registers
        read_ack<='0';
      when Rd_RAM_1 =>
        Infix.Num<='1'; -- infix_1 will read the data
        read_ack<='1'; -- the data have been read, this signal is valid only one clock,
        -- its validation will set the ready_RAM signal
      when others =>
        null;
    end case;
  end if;
end process DATA_RAM_PROC;

----------------- calculate the next state for library management ------------------
SS_NEXT_Lib_PROC: process(ss_Lib,Chk_wt,Cnsl.Cnt,infix_Reg,read_ack,ready.Reg,ready.infix,ready.suffix,DynP_done,
Lib_idle,Rd_Lib_en)
begin
  ss.next_Lib<=ss_Lib;
  case ss_Lib
  is
    when IDLE_Lib =>
      if Chk_wt=INT.WAIT'HIGH then
        ss.next_Lib<=CALCU;
      end if;
    when CALCU =>
      if Cnsl.Cnt='1' then
        ss.next_Lib<=FETCH_RAM;
      else
        ss.next_Lib<=FETCH_REG;
      end if;
    when FETCH_RAM =>
      if read_ack='1' then
        ss.next_Lib<=CHECK.WAIT;
      end if;
    when FETCH_REG =>
if ready_Reg='1' then
    ss.next.Lib<=CHECK.WAIT;
end if;
when CHECK.WAIT =>
    if ready.infix='1' and ready.suffix='1' then -- infix data are prepared
        ss.next.Lib<=CHECK;
    end if;
when CHECK =>
    if DynP.done='1' then
        ss.next.Lib<=LIB.WAIT; -- calculate the infix needed for next check after this check
    end if;
when LIB.WAIT =>
    if Lib.idle='0' then
        ss.next.Lib<=IDLE.Lib;
    end if;
when others=>
    null;
end case;
if Rd.Lib.en='1' then -- disable the FSM when fetching library
    ss.next.Lib<=IDLE.Lib;
end if;
end process SS_NEXT_Lib_PROC;

------------------ library management process ----------------------
DATA.Lib_PROC: process(Clk,Rst)
begin
    if Rst='1' then
        ------------------ reset the output ports -----------------------
        -- signal connecting to peripheral circuits
        Rd_RAM_Req<='0'; -- to MMU
        -- signal connecting to datapath
        infix.Reg<='0';
        Infix_shift<='0';
        Suffix.Chk.en<='0';
        DynP.en<='0';
        ----------------- reset the internal signals -------------------
        -- signal connecting to the library
        Chk_result<='0'; Chk.done<='0';
        position2Lib<=0; rotate2Lib<=0;
        score2Lib<='0'; Bases2Lib<=(others=>0);
        -- flag signals for the controller itself
        next_cycle<='0';
        Calcu.Cnt<='0';
        suffix.Chk.en.lock<='1';
        ready.RAM<='0';
        ready.Reg<='0';
        ready.suffix<='0';
        distance<=0;
        Addr_RAM_Reg<=(others=>'0');
    elsif rising_edge(Clk) then
        case ss.Lib is
            when IDLE.Lib =>
                DynP.en<='0';
                ready.suffix<='0';
                -- signal used for checking
                Calcu.Cnt<='0';
                suffix.Chk.en.lock<='1';
            when CALCUL =>
                Calcu.Cnt<='1';
                if Suffix.trans.en.Li='0' then
                    suffix.Chk.en.lock<='0';
                else
                    if Calcu.Cnt='0' then
                        if rotate4Lib <= Cnt.Comp or Tran.ing='0' then -- find out whether the suffix can be
                            fetched in this cycle
                            next_cycle<='1';
                            -- distance from the head of the sequence
                            distance<=Cnt.Rd-position4Lib-1;
                        else
                            next_cycle<='0';
                            distance<=Cnt.Rd-position4Lib;
                        end if;
                    else
                        if position4Lib <= LENG.FIX then
                            infix.Reg<='0';
                            -- informs the Datapath the infix should be read from
                            RAM
                            else
                                infix.Reg<='1';
                            end if;
                        end if;
                    end if;
                end if;
            when FETCH.REG =>
                if position4Lib >= LENG.FIX then -- correct overlapping and overlapped relationship
                    .............
Infix_shift<=position4Lib-LENG_FIX; -- shift amount ranges from 0 to LENG_FIX, one more bit required
else
Infix_shift<=0; -- wrong overlap found, give any suffix to eliminate it
end if;
if next_cycle='0' then
ready_Reg<='1'; -- get infix from current register
ready_RAM<='1'; -- get infix from next register
end if;
if next_cycle='0' then
Infix_shift<=LENG_FIX-(distance mod LENG_FIX); -- shift amount is determined by the distance
Addr_RAM_REG<=to_unsigned((distance)/(LENG_FIX),WIDTH_ADDR); -- there are LENG_FIX bases in one word
Rd_RAM_REQ<='1';
when FETCH_RAM =>
Infix_shift<=LENG_FIX-(distance mod LENG_FIX); -- shift amount is determined by the distance
Addr_RAM_REG<=to_unsigned((distance)/(LENG_FIX),WIDTH_ADDR); -- there are LENG_FIX bases in one word
Rd_RAM_REQ<='1';
when CHECK =>
    ready_Reg<='0'; -- clear two ready signals that disable the "infix.en"
    ready_RAM<='0';
    DynP_en<='1';
    if DynP_done='1' then -- the finish of checking process
        Chk_result<='1';
        Chk_result<='0';
        Chk_result<='0';
        Chk_result<='1';
when LIB_WAIT =>
    Chk_done<='0'; -- one clock enable signal
when others =>
nul;
end case; -- end of state machine
-- generate the enable signals when each cycle finishes
if Tran_ing /= Tran_ing_L1 then
    if Tran_ing='0' then -- falling edge
        Rd_done<='1';
    else
        cycle_done<='1';
    end if;
else
    Rd_done<='0';
cycle_done<='0';
end if;
-- clear the RAM reading request
if read_ack='1' then -- if data is read
    Rd_RAM_REQ<='0';
end if;
-- store the suffix copy for checking usage
if Cnt_Comp=rotate4Lib and Cnt_Comp/=0 and suffix_Chk_en_lock='0' then
    Suffix_Chk_en<='1';
    Suffix.Chk.en.lock<=1;
    ready.suffix<=1';
else
    Suffix.Chk.en<='0';
end if;
-- one clock delay
Suffix.tran.en.L1<=Suffix.tran.en;
Tran.en.L1<=Tran_ing;
end if; -- end of clock
end process DATA_Lib_PROC;
ready_infix <= ready_RAM or ready_Reg;
Infix.en <= ready_infix;
Rd_seed<=Chk_done or Rd_Lib_en;
--------------------------------------------------------------------------------
-- map the output from library
position4Lib <= to.integer(unsigned(seed4Lib(31 downto 32-WIDTH_READ)));
rotate4Lib <= to.integer(unsigned(seed4Lib(31-WIDTH_READ downto 32 WIDTH_READ-WIDTH_PIXEL)));
BASE_ASSIGN: for i in 0 to REDUN*2 generate
    bases4Lib(i) <= to_integer(unsigned(seed4Lib((i+1)*WIDTH_EdDi-1 downto i*WIDTH_EdDi)));
end generate BASE_ASSIGN;

Bases2DP<=Bases4Lib;
Addr_RAM<=Addr_RAM_Reg;
--Overlap_o<=to_unsigned(position4Lib,WIDTH_READ) & to_unsigned(rotate4Lib,WIDTH_PIXEL);

Infix_from_Reg<=infix_Reg;
end RTL;

Listing D.12: ALU
signal diff: INT_DIFF;
-- signal used to annotate the ports of calculation elements
signal corner,left,up,result: VECT_INT_EDDI(WIDTH_RB*2-2 downto 0);
signal string.V,string.H: SEQ.DNA(WIDTH_RB*2-2 downto 0);

begin
-- clocked process
Cal: process(Clk,Comp_en)
variable diff: INTEGER;
variable coordinate: INTEGER;
variable distance_v : INTEGER;
begin
if rising_edge(Clk) then
if Comp_en='0' then
-- reset output signal
Comp_done <='0';
-- reset internal signal
wave_Reg <=(others=>0);
shift_Reg_H<=(others=>'00');
shift_Reg_V<=(others=>'00');
else
if Cal_Cnt < INT_DP'HIGH then
Cal_Cnt <= Cal_Cnt+1;
else
Comp_done <='1';
end if;
-- shift the assign register
if (Cal_Cnt mod 2) = 1 then
shift_Reg_H(shift_Reg_H'HIGH downto 1) <= shift_Reg_H(shift_Reg_H'HIGH-1 downto 0);
shift_Reg_V(shift_Reg_V'HIGH downto 1) <= shift_Reg_V(shift_Reg_V'HIGH-1 downto 0);
end if;
-- initialization of wave_Reg register when counter equals to zero
if Cal_Cnt=0 then
wave_Reg(WIDTH_RB*2-DIST_MAX-1 downto DIST_MAX-1) <= Bases_i;
BASE_H: for i in 1 to DIST_MAX-1 loop
distance_v := Bases_i(REDUN*2) + i;
if distance_v < DIST_MAX then
wave_Reg(i+WIDTH_RB*2-DIST_MAX-1) <= distance_v;
else
wave_Reg(i+WIDTH_RB*2-DIST_MAX-1) <= DIST_MAX;
end if;
distance_v := Bases_i(0) + i;
if distance_v < DIST_MAX then
wave_Reg(DIST_MAX-1-i) <= distance_v;
else
wave_Reg(DIST_MAX-1-i) <= DIST_MAX;
end if;
end loop BASE_H;
-- initialize the shifting NT assign register
shift_Reg_H(LENG_FIX-1 downto 0) <= Word_H;
shift_Reg_V(LENG_FIX-1 downto 0) <= Word_V;
else
-- Cal_Cnt is larger than 0
-- selective enable register update
for i in 0 to WIDTH_RB*2-2 loop
if (i mod 2)=Cal_Cnt_L1 mod 2 and abs(i-WIDTH_RB+2)=Cal_Cnt_L1 then
-- registers have different parity with Cnt
wave_Reg(i) <= result(i);
end if;
end loop;
end if;
-- end of clock
end process Cal;

-- process to find the minimum number and store the bases, which is the Block Connection Shift algorithm
BCS: process(Clk,Comp_en)
variable min.V: INT_EDDI :=INT_EDDI'HIGH;
variable min.H: INT_EDDI :=INT_EDDI'HIGH;
variable shift.H: INT_SHIFT :=0;
variable shift.V: INT_SHIFT :=0;
begin
if rising_edge(Clk) then
if Comp_en='0' then
min.V :=INT_EDDI'HIGH;
min.H :=INT_EDDI'HIGH;
shift_H :=0;
shift_V :=0;
else
-- find minimum number and store the bases
if BCS.H < min.H then
min.H <= BCS.H;
shift.H <= Cal.Cnt.L;
else
if BCS.V < min.V then
min.V <= BCS.V;
shift.V <= Cal.Cnt.L;
else
end if;
end if;
end if;
end process Cal;
min_Reg <= 0;
shift <= 0;
base_C <= 0;
base_V <= (others=>0);
base_H <= (others=>0);
else
  diff <= LENG_FIX*2-3-REDUN-Cal_Cnt;
  if diff=0 and diff=REDUN then -- cycles for checking the minimum number
    -- find the new minimum number for row and column separately
    if wave_Reg(WIDTH_RB-1-diff) <= min_H then
      min_H <= wave_Reg(WIDTH_RB-1-diff);
      shift_H:= -diff;
    end if;
    if wave_Reg(WIDTH_RB-1-diff) <= min_V then
      min_V <= wave_Reg(WIDTH_RB-1-diff);
      shift_V:= -diff;
    end if;
    -- determine the shift amount
    min_Reg<=min_V; -- assign the minimum value
    if min_H < min_V then
      shift<=shift_H;
    elsif min_H > min_V then
      shift<=shift_V;
    elsif min_H = min_V then
      shift<=0;
    end if;
    -- record the corner data
    base_C<=wave_Reg(WIDTH_RB-1+shift);
  end if; -- end of checking number
  for i in 0 to REDUN-1 loop
    if abs(shift)+Cal_Cnt=LENG_FIX*2-1-REDUN then
      base_V(i)<=wave_Reg(WIDTH_RB-1+i+1+shift);
      base_H(i)<=wave_Reg(WIDTH_RB-1-i-1+shift);
    end if;
  end loop;
  -- generate the output
  if Cal_Cnt=INT_DP'HIGH then
    bases_o(REDUN) <= base_C-min_Reg;
    for i in 0 to REDUN-1 loop
      bases_o(REDUN+1+i)<= base_V(i)-min_Reg;
    end loop;
    for i in 0 to REDUN-1 loop
      bases_o(REDUN-1-i)<= base_H(i)-min_Reg;
    end loop;
  end if;
end if; -- end of enable
end if; -- end of clock;
end process BCS;
-- build the connection between calculation elements and two input strings
NT_ASSIGN: process(shift_Reg_H,shift_Reg_V)
begin
  for i in WIDTH_RB*2-2 downto 0 loop
    string_H(i)<=shift_Reg_H(shift_Reg_H'HIGH-(string_H'HIGH-i)/2);
    string_V(i)<=shift_Reg_V(shift_Reg_V'HIGH-i/2);
  end loop;
end process NT_ASSIGN;
-- instantiation of calculation elements
ELE: for i in WIDTH_RB*2-2 downto 0 generate
  element: TABLELEMENT port map(
    Xin=>String_V(i),
    Yin=>String_H(i),
    Corner=>corner(i),
    Left=>left(i),
    Up=>up(i),
    Dist=>result(i));
end generate ELE;
-- build the connection between calculation elements and wave_Reg register
CAL_REG: for i in 1 to WIDTH_RB*2-3 generate
  corner(i) <= wave_Reg(i);
  left(i) <= wave_Reg(i-1);
  up(i) <= wave_Reg(i-1);
end generate CAL_REG;
corner(WIDTH_RB*2-2) <= wave_Reg(WIDTH_RB*2-2);
Listing D.13: Overlap Library

```vhdl
-- FILE name: Lib_Ovlap.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 15/09/2012
-- version: 4.0
-- Abstract: This file is the library used to store the potential overlaps. Each overlap contains 5 parts:
-- 1. rotate: the relative distance between two matching reads
-- 2. position: the length of detected read when the overlap is found
-- 3. score: the accumulated scores during the whole process
-- 4. bases: base numbers for connecting two blocks
-- 5. chk wt: cycles to be waited before the next check, to avoid to repeat the errors
-- 6. lock: when the lock symbol is active, no writing process is allowed
-- The whole library pool is implemented like a shift register array, the new seeds are always
-- written from the top of the array, and shifted towards the bottom until the next location is locked.
-- Some abbreviations: Comp=compare, Lib=library, Chk=check, wt=wait, Ovlap=overlap
-- Called by: LOCAL_CTRL.vhd
-- Revision history: v3.5

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;

entity Lib_Ovlap is
  port(
    Clk,Rst: in STD_LOGIC;
    -- signal from global controller
    Cnt_Rd: in INT_RD;
    Cnt_Comp: in INT_PIXEL_ALL;
    Matching: in STD_LOGIC;
    -- signal from intra-pixel
    Cycle_done: in STD_LOGIC;
    Rd_seed: in STD_LOGIC;
    Chk_result in STD_LOGIC;
    -- data to store
    Position_i: in INT_RD;
    Rotate_i: in INT_PIXEL_ALL;
    Score_in: in INT_EdDi;
    Bases_in: in VECT_INT_EdDi(REDUN*2 downto 0);
    -- signal to local pixel
    Lib_idle: out STD_LOGIC;
    Seed_o: out OVERLAP;
    Chk_Wt_o: out INT_WAIT);
end Lib_Ovlap;

architecture RTL of Lib_Ovlap is
  component fifo_32 port(
    clock : IN STD_LOGIC;
    data : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    rdreq : IN STD_LOGIC;
    wrreq : IN STD_LOGIC;
    q : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
  );
end component;

type VECT_INT_WAIT is array (NATURAL range <>) of INT_WAIT;
signal Cnt_vect_Wt: VECT_INT_WAIT(LENG_FIX-1 downto 0);
signal Cnt_seed: UNSIGNED(3 downto 0); -- the depth of fifo
signal cycle_done_L1: STD_LOGIC;
signal Chk_result_L1: STD_LOGIC;
signal data_sig: STD_LOGIC_VECTOR(31 DOWNTO 0);
signal wr_en: STD_LOGIC;
begin
```

```
fifo_inst : fifo_32 PORT MAP (  
clock => Clk,  
data => data_sig,  
rqreq => Rd_seed,  
wrreq => wr_en,  
q => Seed_o  
);  

--INPUTMUX: process  
MMU: process(Rst,Clk)  
variable data_32: UNSIGNED(31 downto 0);  
variable bases_raw: UNSIGNED((REDUN*2+1)*WIDTH_EdDi-1 downto 0);  
begin  
if Rst='1' then  
data_sig <= (others=>'1');  
Cnt_vect_Wt <= (others=>0);  
Cnt_seed <= (others=>'0');  
cycle_done_L1 <= '0';  
Chk_result_L1 <= '0';  
elsif rising_edge(Clk) then  
--------------------------------- read information into seed --------------------------------  
-- record the new seed from the exact matching  
if Matching='1' then  
  for i in 0 to REDUN-1 loop  
    bases.raw((i+1)*WIDTH_EdDi-1 downto i*WIDTH_EdDi) := to_unsigned(abs(i-REDUN),WIDTH_EdDi);  
  end loop;  
  data_32 := to_unsigned(Cnt_Rd,WIDTH_READ) & to_unsigned(Cnt_Comp,WIDTH_PIXEL) & to_unsigned(0,WIDTH_ERROR) & bases.raw;  
  if Chk_result='1' then -- two writing process happen at the same cycle  
    Chk_result_L1 <= '1';  
  end if;  
  elsif Chk_result='1' or Chk_result_L1='1' then -- add the checked seeds back  
    for i in 0 to REDUN+2 loop  
      bases.raw((i+1)*WIDTH_EdDi-1 downto i*WIDTH_EdDi) := to_unsigned(Bases_in(i),WIDTH_EdDi);  
    end loop;  
    data_32 := to_unsigned(Position_i,WIDTH_READ) & to_unsigned(Rotate_i,WIDTH_PIXEL) & to_unsigned(Score_in,WIDTH_ERROR) & bases.raw;  
    Chk_result_L1 <= '0';  
  end if;  

------------------ shift the waiting counts when there is a checking ------------  
if Rd_seed='1' then  
  Cnt_seed:=Cnt_seed-1;  
  for i in 0 to Cnt_vect_Wt’HIGH-1 loop  
    Cnt_vect_Wt(i) <= Cnt_vect_Wt(i+1);  
  end loop;  
  Cnt_vect_Wt(Cnt_vect_Wt’HIGH)<=0;  
  if Cycle_done='1' then  
    cycle_done_L1 <= '1';  
  end if;  
  elsif Cycle_done='1' or cycle_done_L1='1' then  
    count the waiting cycles  
    for i in Cnt_vect_Wt’range loop  
      if i=Cnt.seed and Cnt_vect_Wt(i)=INT.WAIT’HIGH then  
        Cnt_vect_Wt(i) <= Cnt_vect_Wt(i) + 1;  
      end if;  
    end loop;  
    cycle_done_L1 <= '0';  
  end if;  
  if Matching='1' or Chk_result='1' or Chk_result_L1='1' then  
    data_sig <= std_logic_vector(data_32);  
    wr_en <= '1';  
    if Cnt seed=15 and Rd_seed='0' then -- the depth of the fifo  
      Cnt.seed:=Cnt.seed-1;  
      elsif Rd.seed='1' then  
        Cnt.seed:=Cnt.seed;  
        elsif Rd.seed='0' then  
          wr_en<='0';  
        end if;  
    else  
      Lib_idle <= Matching or Chk_result or Chk_result_L1 or Cycle_done or cycle_done_L1 or Rd_seed;  
      Chk.Wt.o <= Cnt_vect_Wt(0);  
    end if;  
  end if;  
end process MMU;  
end RTL;
-- FILE name: Tablelement.vhd
-- Author: Yuanqi Hu
-- Email: yuanqi.hu09@imperial.ac.uk
-- Date: 24/08/2011
-- version: 1.0
-- Abstract: This file describes the dynamic programming algorithm in one location. It has three previous distance
-- values: Corner, Left and Up. The increment is determined by two data input, Xin and Yin. And the new distance
-- value is obtained via selecting the smallest value through these three candidates.
-- Called by: ALU.vhd
-- Revision history: N/A

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use WORK.PARAMETERS.all;

entity TABLELEMENT is
port(
    Xin: in BASE;
    Yin: in BASE;
    Corner: in INT.EdDi;
    Left: in INT.EdDi;
    Up: in INT.EdDi;
    Dist: out INT.EdDi
);
end TABLELEMENT;

architecture COMBINATIONAL of TABLELEMENT is
begin
    CAL: process(Xin,Yin,Corner,Left,Up)
    variable corner_new,left_new,up_new: INT.EdDi;
    begin
        ----- calculate the new values after incrementaion-------------------
        if Corner<INT.EdDi'HIGH and Xin/=Yin then
            corner_new:=Corner+1;
        else
            corner_new:=Corner;
        end if;
        if Left<INT.EdDi'HIGH then
            left_new:=Left+1;
        else
            left_new:=Left;
        end if;
        if Up<INT.EdDi'HIGH then
            up_new:=Up+1;
        else
            up_new:=Up;
        end if;
        ----------- select the smallest one ----------------------
        Dist:=corner_new;
        if corner_new>left_new or corner_new>up_new then
            if left_new>up_new then
                Dist:=up_new;
            else
                Dist:=left_new;
            end if;
        end if;
        end process CAL;
    end COMBINATIONAL;
Appendix E

Codes of Control logic in the Sensing system

Listing E.1: Control logic in the Sensing system

```vhdl
library IEEE;
useIEEE.std_logic_1164.all;
useIEEE.numeric_std.all;

-- the main difference of v3 is that the address is low effective
entity spi_A10D8_v4 is
  generic(
    BITS_ADDR : INTEGER := 10 -- 32x32
  );
  port(
    -- signal from chip
    Clk: in STD_LOGIC;
    Rst: in STD_LOGIC;
    Data_ready: in STD_LOGIC;
    Sel_R: in UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
    Sel_C: in UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
    Data: in UNSIGNED(7 downto 0);
    -- signal to off-chip fpga
    MOSI: out STD_LOGIC;
    Rd.en: out STD_LOGIC
  );
end entity spi_A10D8_v4;

architecture rtl of spi_A10D8_v4 is
  type MULT_STATE is (IDLE,TRANS,STORE);
  signal ss,ss_next: MULT_STATE;
  signal Addr: UNSIGNED(BITS_ADDR-1 downto 0);
  signal reg: UNSIGNED(BITS_ADDR+7 downto 0); -- total length = the sum of address and data
  signal Cnt_tran: UNSIGNED(4 downto 0); -- count 18 cycles
  begin
    ENCODER: process(Sel_C,Sel_R)
    begin
      Addr<=(others=>'0');
      for i in 0 to 2**(BITS_ADDR/2)-1 loop
        if Sel.R(i)='0' then -- this number is the selected
          Addr(BITS_ADDR-1 downto BITS_ADDR/2) <= to_unsigned(i,BITS_ADDR/2);
        end if;
        if Sel.C(i)='0' then -- this number is the selected
          Addr(BITS_ADDR/2-1 downto 0) <= to_unsigned(i,BITS_ADDR/2);
        end if;
      end loop;
    end process ENCODER;
    SS_PROC: process(Clk,Rst)
    begin
      if Rst='1' then
        ss<=IDLE;
      elsif rising_edge(Clk) then
```
ss<=ss.next;
end if;
end process SS.PROC;

------------ calculate the next state for reading the data from RAM --------------

SS_NEXT_PROC: process(ss,Data_ready,Cnt_tran)
begin
ss.next<=ss;
case ss is
when IDLE =>
if Data_ready='1' then
ss.next<=STORE;
end if;
when STORE =>
ss.next<=TRANS;
when TRANS =>
if Cnt_tran=BITS_ADDR+7 then
ss.next<=IDLE;
end if;
when others =>
NULL;
end case;
end process SS_NEXT_PROC;

PROC: process(Clk,Rst)
begin
if Rst='1' then
Cnt_tran<=(others=>'0');
MOSI<='0';
Rd_en<='0';
reg<=(others=>'0');
elsif rising_edge(Clk) then
case( ss ) is
when IDLE =>
Rd_en<='0';
Cnt_tran<=(others=>'0');
when STORE =>
reg(BITS_ADDR+7 downto 0) <= Addr;
when TRANS =>
MOSI<='1';
reg(BITS_ADDR+7 downto 1) <= reg(BITS_ADDR+6 downto 0);
end
when others =>
NULL;
end case;
end if; end of clock
end process PROC;
end rtl;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity decoder_bar is
   generic(
      BITS_ADDR: INTEGER := 10
   );
   port(
      Ain: in UNSIGNED(BITS_ADDR-1 downto 0);
      Sel_R:out UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
      Sel_C:out UNSIGNED(2**(BITS_ADDR/2)-1 downto 0)
   );
end decoder.bar;

architecture rtl of decoder.bar is
begin
process(Ain)
variable Cnt.R : UNSIGNED((BITS_ADDR/2)-1 downto 0);
variable Cnt.C : UNSIGNED((BITS_ADDR/2)-1 downto 0);
begin
Cnt.C := Ain(BITS_ADDR/2-1 downto 0);
Cnt.R := Ain(BITS_ADDR-1 downto BITS_ADDR/2);
Sel.R <= (others=>'1');
Sel.C <= (others=>'1');
Sel.R(to.integer(Cnt.R)) <= '0';
Sel.C(to.integer(Cnt.C)) <= '0';
end process;
end rtl;
-- main feature of v6 is there is no internal memory to store offset information. do the real time SPI
-- with external memory block.
-- only change in v7 is that the select signal is low effective

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

dc entity digi cds vX.32x32 is

port(
  Clk,Rst: in STD_LOGIC;
  Cds_en: in STD_LOGIC;
  Sense_en: in STD_LOGIC;
  Turbo: in STD_LOGIC;
  Sel.R_o: out UNSIGNED(31 downto 0);
  Sel.C_o: out UNSIGNED(31 downto 0);

  Bin.os: in UNSIGNED(7 downto 0);
  EOC_cds: in STD_LOGIC;
  EOC_sense: in STD_LOGIC;
  ADC_cds_start: out STD_LOGIC;
  ADC_sense.start: out STD_LOGIC;
  Dout: out UNSIGNED(7 downto 0);

  Sel.R: in STD_LOGIC;
  Sel.C: in STD_LOGIC;
  Rd.cds.req: out STD_LOGIC;
  SO_en_L1: out STD_LOGIC;
  SI_os: out STD_LOGIC;
  SI_en: in STD_LOGIC;
  SI_os_L1: out STD_LOGIC;
  SI_en_L1: out STD_LOGIC;

  Dout_final: in UNSIGNED(7 downto 0);
  Data_out_en: out STD_LOGIC;
  MOSI: out STD_LOGIC;

  -- signal connecting with external memory block
  SI_os: in STD_LOGIC;
  SI_en: in STD_LOGIC;
  Rd.cds.req: out STD_LOGIC;
  SO_en_L1: out STD_LOGIC;
  SO_os: out STD_LOGIC)
);

end entity digi cds vX.32x32;

architecture rtl of digi cds vX.32x32 is

constant NUM_PIXEL: INTEGER := 1024;
constant BITS_ADDR: INTEGER := 10;
constant BITS_ADC: INTEGER := 8;

component decoder_bar is
generic(
  BITS_ADDR: INTEGER := 6
);
port(
  Ain: in UNSIGNED(BITS_ADDR-1 downto 0);
  Sel.R: out UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
  Sel.C: out UNSIGNED(2**(BITS_ADDR/2)-1 downto 0)
);
end component;

component spi A10D8_v4 is

generic(
  BITS_ADDR: INTEGER := 10 -- 32x32
);
port(
  Clk: in STD_LOGIC;
  Rst: in STD_LOGIC;
  Data_ready: in STD_LOGIC;
  Sel.R: in UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
  Sel.C: in UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
  Data: in UNSIGNED(7 downto 0);
  MOSI: out STD_LOGIC;
  Rd_en: out STD_LOGIC
);
end component;

struct MULTI_STATE_CDS is
  IDLE, INITIAL, CDS, SENSE, SETTLE, HOLDING;
end struct;

struct MULTI_STATE_SPI is
  IDLE, TRANS, RECE, DRIVE;
end struct;

type MULTI_STATE_CDS is
  (IDLE, INITIAL, CDS, SENSE, SETTLE, HOLDING);

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
signal Cnt.CDS_SPI: UNSIGNED(3 downto 0);

signal Sel.R: UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);
signal Sel.C: UNSIGNED(2**(BITS_ADDR/2)-1 downto 0);

signal Cnt.Initial: UNSIGNED(4 downto 0);
signal Cnt.settle: UNSIGNED(4 downto 0);
signal Csettle: UNSIGNED(4 downto 0);
signal Cnt.ADC: UNSIGNED(3 downto 0);
signal Addr.curr: UNSIGNED(BITS_ADDR-1 downto 0);
signal Addr.out.s: UNSIGNED(BITS_ADDR-1 downto 0);

begin
SETTLECYCLE: process(Turbo)
begin
if (Turbo='1') then
  Csettle<="01000";
else
  Csettle<="11111";
end if;
end process SETTLECYCLE;

---------------------- states transferring -----------------------------
SS_PROC: process(Clk,Rst)
begin
if Rst='1' then
  ss<=IDLE;
elsif rising_edge(Clk) then
  ss<=ss_next;
  ss_spi<=ss_next_spi;
end if;
end process SS_PROC;

SS_PROC_SPI: process(ss_spi,EOC_cds,SI_en,Cnt.CDS_SPI)
begin
ss_next_spi<=ss_spi;
case ss_spi is
when IDLE =>
  if EOC_cds='1' then
    ss_next_spi<=TRANS;
  end if;
when TRANS =>
  if Cnt.CDS_SPI="0111" then
    ss_next_spi<=IDLE;
  end if;
when RECE =>
  if SI_en='0' then
    ss_next_spi<=DRIVE;
  end if;
when DRIVE =>
  ss_next_spi<=IDLE;
when others =>
  NULL;
end case;
end process SS_PROC_SPI;

------------ calculate the next state for reading the data from RAM -----------
SS_NEXT_PROC: process(ss,Cds.en,Sense.en,Cnt.Initial,EOC_cds,EOC_sense,Cnt.settle,Addr.out.s)
begin
ss.next=ss;
case ss is
when IDLE =>
  if Cds.en='1' or Sense.en='1' then
    ss.next=INITIAL;
  end if;
when INITIAL =>
  if Cnt.Initial = Csettle then
    if Sense.en='0' then
      ss.next=CDS;
    else
      ss.next=SENSE;
    end if;
  end if;
when CDS =>
  if EOC_cds = '1' then  -- end of the offset sampling of one pixel
    if Addr.out.s=(BITS_ADDR-1 downto 0 => '0') then
      ss.next=HOLDING;
    else
      ss.next=IDLE;
    end if;
  end if;
when OVR =>
  ss.next=RECE;
when OUTOF ->
  ss.next=DRIVE;
when others =>
  NULL;
end case;
end process SS_NEXT_PROC;
ss.next<=SETTLE;
end if;
end if;
when SENSE =>
if EOC_sense = '1' then
  if Sense_en='0' then
    ss.next<=IDLE;
  else
    ss.next<=SENSE;
  end if;
end if;
when SETTLE =>
if Cnt_settle=Csettle then
  if Sense_en='0' then
    ss.next<=CDS;
  else
    ss.next<=SENSE;
  end if;
end if;
when HOLDING =>
if Cds_en='0' then
  ss.next<=IDLE;
end if;
when others =>
  NULL;
end case;
end process SS_NEXT_PROC;

DATA_PROC: process(Clk,Rst)
begin
if Rst='1' then
  -- internal signals
  Cnt_ADC<=(others=>'0');
  Cnt_settle<=(others=>'0');
  Cnt_Inital<=(others=>'0');
  -- Addr_curr<=(others=>'0');
  Addr_out_s<=(others=>'0');
  Addr_curr<=(others=>'0');
  Rd_cds_req<='0';
  -- output signals
  ADC_cds_start<='0';
  ADC_sense_start<='0';
elsif rising_edge(Clk) then
  case ss is
  when IDLE =>
    -- disable the read enable signal
    Cnt_ADC<=(others=>'0');
    Rd_cds_req<='0';
    Addr_out_s<=(others=>'0');
    Cnt_Inital<=Cnt_Inital+1;
  when INITIAL=>
    -- disable the read enable signal
    Cnt_ADC<=(others=>'0');
    Rd_cds_req<='0';
    Addr_out_s<=(others=>'0');
    Cnt_Inital<=Cnt_Inital+1;
  when CDS =>
    -- enable the ADC process
    ADC_cds_start<='1';
  when SENSE =>
    -- reset previous signals
    Rd_cds_req<='0';
    Cnt_ADC<=(others=>'0');
    -- enable the ADC process
    ADC_sense_start<='1';
  when others =>
    NULL;
  end case;
end if;
end process DATA_PROC;
-- calculate the real output
end if;
when SETTLE =>
    -- reset the previous enable signals
    Cnt_settle<=Cnt_settle+1;
    if Cnt_settle=Csettle then
        Cnt_settle<=(others=>'0');
    end if;
when HOLDING =>
    NULL;
when others=>
    NULL;
end case;
end if; -- end of clock
end process DATA_PROC;

CDS_SPI: process(Rst,Clk)
begin
    if (Rst='1') then
        Cnt_CDS_SPI<=(others=>'0');
        Reg_cds<=(others=>'0');
        SO_en<='0';
        Dout<=(others=>'0');
    elsif rising_edge(Clk) then
        case ss_spi is
            when IDLE =>
                Cnt_CDS_SPI<=(others=>'0');
                SO_en<='0';
            when RECE =>
                Reg_cds(0) <= SI_os_L1;
                Reg_cds(BITS_ADC-1 downto 1)<=Reg_cds(BITS_ADC-2 downto 0);
            when DRIVE =>
                Dout<=Reg.cds;
            when TRANS=>
                Cnt_CDS_SPI<=Cnt_CDS_SPI+1;
                SO_en<='1';
                if Cnt.CDS_SPI="0000" then
                    Reg.cds<=Din.os;
                    else
                        Reg.cds(BITS_ADC-1 downto 1)<=Reg.cds(BITS_ADC-2 downto 0);
                end if;
            when others =>
                NULL;
        end case;
    end if;-- end of clock
end process CDS_SPI;

SPI_INST: spi_A10D8_v4
generic map( 
    BITS_ADDR => BITS_ADDR
)
port map( 
    -- signal from chip
    Clk=>Clk, 
    Rst=>Rst, 
    Data_ready=>EOC_sense, 
    Sel_R=>Sel_R, 
    Sel_C=>Sel_C, 
    Data=>Din_final,
    -- signal to off-chip fpga
    MOSI=>MOSI, 
    Rd_en=>Data_out_en
);

DECODER_INST: decoder_bar
generic map( 
    BITS_ADDR => BITS_ADDR
)
port map( 
    Ain=>Addr.out.s, 
    Sel_R=>Sel_R, 
    Sel_C=>Sel_C
);

Sel_R_o<=Sel_R;
Sel_C_o<=Sel_C;
end rtl;
Appendix F

PCB test circuits
Figure F.1: Schematic of test circuits PCB
Figure F.2: Layout of test circuits PCB
Appendix G

pH buffer Preparation

1. Take a 1 litre plastic bottle, add a magnetic stirrer into the bottle (for proper mixing).
2. To make 100mM final concentration, add 100ml of 1M Tricine (stock concentration)
3. Then add 333.33ml of 3M KCL solution to make the final working concentration to 1M into the above plastic bottle.
4. Fill the bottle with sigma /double distilled water to 1 Litre as a final volume.
5. Place the bottle on the stirrer for mixing.
6. Place the pH probe into the bottle and adjust the pH as required by NaOH or HCL (for higher pH 7-10 use 10M NaOH)