Design and Simulation of a Readout Circuit for Capacitive Sensing in CMOS-MEMS Devices

by

Ahmad Jazli bin Abdul Rahman
14759

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics)

JANUARY 2015

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Design and Simulation of a Readout Circuit for Capacitive Sensing in CMOS-MEMS Devices

by

Ahmad Jazli bin Abdul Rahman
14759

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS)

Approved by,

____
(AP DR JOHN OJUR DENNIS)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

AHMAD JAZLI BIN ABDUL RAHMAN

ABSTRACT

Many electronic devices have started to pay attention to CMOS-MEMS technology which provides many advantages. Mostly used in accelerometer and gyroscopes, CMOS-MEMS provides a lot of challenges including the sensing of the output signal. While the signal is about femto- (10^{-15}) to atto- (10^{-18}) farad, the noise presence in the signal is also another constraint. One of the method in sensing the signal is Capacitive Sensing which utilize the difference of the capacitance and distance produced between the comb fingers of capacitances. The front-end circuit is very crucial in producing clean signal and free from noises. This paper use voltage sensing technique of the front-end circuit to evaluate the signal. The evaluation is done by using two techniques; circuit simulation by using MultiSim software, and mathematical modeling by using MATLAB. The result obtained shows that the circuit with Chopper Stabilization technique used has much better output than the circuit without the technique used.

ACKNOWLEDGEMENT

In the name of Allah, The Most Gracious and The Most Merciful. Alhamdulillah, all praises to Allah for giving the blessing and precious opportunity to complete my Final Year Project. Without His assistance, it would be impossible for me to successfully achieve the expected outcome.

I would like to give my gratitude to everyone who had contributed and playing important roles in making my Final Year Project a success. First of all, I would like to thank my family especially my parents who has been giving the support throughout the completion of the project.

I also want to thank the Department of Electrical & Electronics Engineering of Universiti Teknologi PETRONAS for opening the opportunity for me to gain experience in the research area. Other than that, I would like to give special gratitude to Dr John Ojur Dennis acting as my Final Year Project Supervisor for his help in guiding me for the project, monitor my progress, as well as answer the questions from my side despite his busy works and schedule. His advices, encouragements, and patience would be very precious throughout my project completion as well as my future.

Once again I would like to give a very big thank you to those who has giving me an unforgettable journey and meaningful knowledge and advices throughout the completion of my Final Year Project, directly or indirectly.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1:

INTRODUCTION

Complementary metal-oxide-semiconductor (CMOS) has proven to be one of the main components of the today's advance world. It's characteristics of having low required but with high performance has seen it contribution in a lot of electronics components, especially those with microcontrollers, microprocessors, and various digital logic circuits (Borkar, 2006). The researchers keep on finding way to improve CMOS to be as efficient and effective as possible, and one of the technology found is by integrating it with MicroElectroMechanical Systems (MEMS).

MEMS is a process technology that integrate together the mechanical and electrical into components (*An Introduction to MEMS*, 2002). Discovered in the early 1990's, MEMS have the ability to sense and react to the changes in its surrounding through microcircuit control (Varadan, Vinoy, and Gopalakrishnan, 2006). This allow MEMS to be used in many application including computer system, automotive, navigation, sports and health care (Yazdi, Ayazi, and Najafi, 1998). Some examples of the application that utilized MEMS as the mechanism are automotive airbag sensor, medical pressure sensor, and inkjet printer head.

Even though both CMOS and MEMS have shown a lot of advantages on their own, integration of both of them has shown a lot of improvement in the performances, smaller packaging, and lower cost (Witvrouw, 2006). The most common application that used CMOS-MEMS integration technology are accelerometers, gyroscopes, and micro scanners (Xie and Fedder, 2002).

1.1 BACKGROUND

The performance of a circuit is always required to analyze and further improve the system. In CMOS-MEMS, to evaluate the performance, the output of the system is detected and read out which can be done by using several methods. The researchers usually used the Capacitive Sensing method since it has advantages like low noise and power dissipation, low-temperature sensitivity, and its ability to compatible with VLSI technology scaling (Wu, Fedder, and Carley, 2004).

In this method, two sets of comb-like-capacitors are arranged facing each other with some distance between them, as can be seen in Figure 1. One of the capacitors will be set to be fixed while the other will be moving up and down. When there are changes in the distance between the capacitors, the value of capacitance will be changed depending on the magnitude and direction of the moving capacitances. These changes will be detected and translated into capacitance value which will be read by a readout circuits.

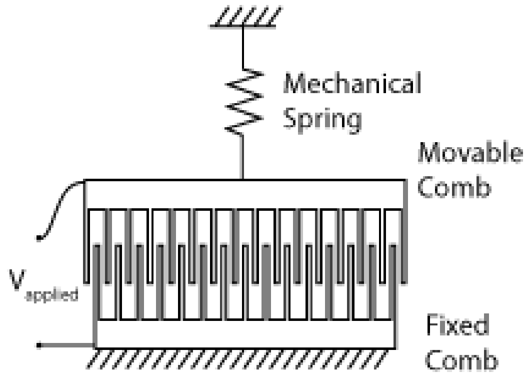


Figure 1 Comb Finger Capacitor

The relationship between the distance and the capacitance can be seen in Figure 2, 3 and 4.

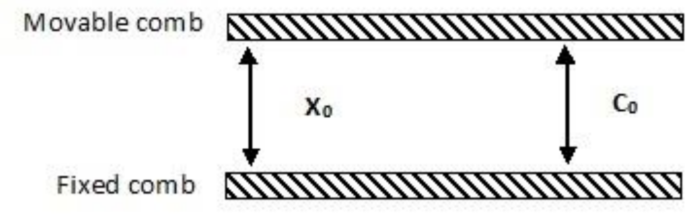


Figure 2 Capacitance Initial Condition

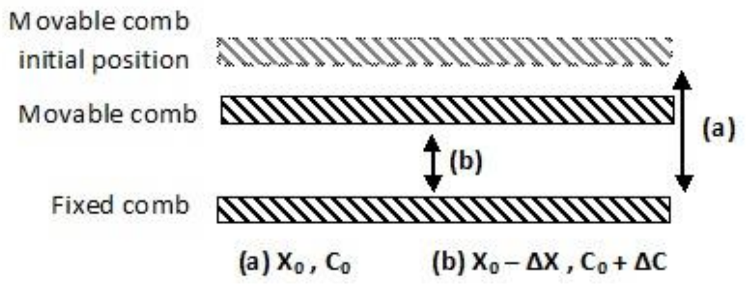

Movable comb initial position
Movable comb
(a)
(b)
Fixed comb
(a) X_0 , C_0
(b) $X_0 - \Delta X$, $C_0 + \Delta C$

Figure 3 Capacitance Moving Condition (Downward)

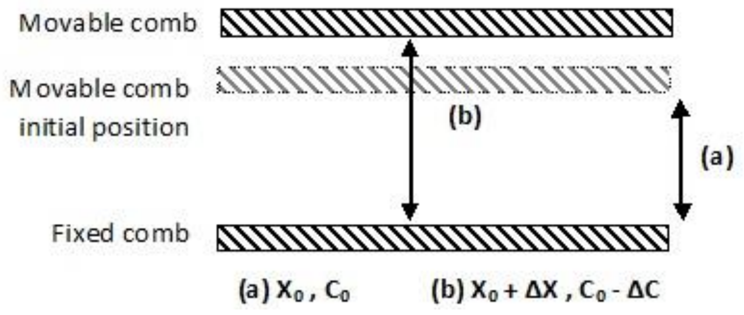

Movable comb
Movable comb initial position
(b)
(a)
Fixed comb
(a) X_0 , C_0
(b) $X_0 + \Delta X$, $C_0 - \Delta C$

Figure 4 Capacitance Moving Condition (Upward)

In figure 2, the capacitance at initial condition where no input signal is given or detected by the circuit. At this stage, the displacement and capacitive value is at their initial value too (X0 and C0 respectively). But when a signal is sensed, the movable comb will start to move. This can be seen in Figure 3 and 4 where the movable comb is either moving downward and upward, depending on the signal it received. This will change the value of the displacement and also affected the capacitive value. When the capacitance moving downward, the displacement will decreasing (X0-ΔX) but the capacitive value increasing (C0+ΔC), while for the upward motion, their values are vice versa (displacement increasing, X0+ΔX, Capacitive decreasing, C0-ΔC). This change in capacitive value that will give the input to the sensing device, which in this case, the capacitive sensing circuit.

1.2 PROBLEM STATEMENT

Advanced technology always came with challenging constraints. In these early stage of developing CMOS-MEMS devices, several challenges pose by this technology. One of the major problem with CMOS-MEMS applications is the challenge to read the output signal which is very small, ranging from femto- (10^{-15}) to atto- (10^{-18}) farad. In addition, since CMOS-MEMS is an electronic components, it cannot run away from the presence of noise in the circuits, which sometimes can be much bigger than the actual signal (up to pico-farad). Without the removal of the noise, it is really difficult to differentiate between the real signal and the unwanted noise.

1.3 OBJECTIVES AND SCOPE OF STUDY

1.3.1 Objectives

In general, this project goal is to develop a capacitive readout circuit that has high Signal-to-Noise Ratio (SNR).

In short, the project aims to:

1. To develop and simulate a mathematical modeling of capacitive readout circuit CMOS-MEMS devices
   A mathematical modeling simulating the parameter chosen will be done through MATLAB software.
2. To simulate the capacitive readout circuit of CMOS-MEMS devices
   After the mathematical modeling has successfully being design, the circuit will be simulated by using MultiSim software to evaluate and improve the performance.

1.3.2 Scope of Study

There are three part of CMOS-MEMS that can be manipulate to achieve the best result, which are sensing circuit design, transducer design and fabrication, and control system design (Fang, 2006). For this work, the focus is the sensing circuit design while the other two parts are out of the scope of this project. There are three sensing circuit architectures for capacitive sensing, namely, Capacitive Sensing, Continuous Time Current (CTC) Sensing, and Continuous Time Voltage (CTV) Sensing. The work will be focusing on the Continuous Time Voltage Sensing architecture but all of the architectures will be discussed briefly. While various parameters available to determine the performance of the CMOS-MEMS circuits, this work will only focusing on achieving the best Signal-to-Noise Ratio (SNR).

1.3.3 Relevancy of the Project Scope

The project is selected to be focusing on the capacitance sensing only due to time constraint of the project itself. The project needs to be completed in the period of 7 months only. However, the capacitance sensing itself has various type of architectures. Thus, the best architecture can be selected based on the literature review done. In addition, there are also various circuit configuration that has been done in other researches. This leaves the project with various choice of circuits to be working with.

CHAPTER 2:

LITERATURE REVIEW AND THEORY

2.1 READOUT CIRCUIT ARCHITECTURES FOR CAPACITIVE SENSORS

Readout circuit for capacitive sensing can be divided into three which are Capacitive Sensing (Charge Sensing), Current Sensing, and Voltage Sensing. Each of the architecture possess their own advantages with some restriction. All three architectures will be discussed in this part.

2.1.1 Capacitive Sensing/ Charge Sensing

Capacitive sensing can arranged in two ways in order to sense the changes in the circuits. The first method is by using capacitive divider as shown in Figure 5. The topology is just the same as half-bridge circuit which the output signals will be sensed at the center of the sensing capacitance node. These sensing capacitances are powered by modulation voltage. The second topology is fully differential capacitive bridge. The circuit will sensed the output signal at the central node just the same as capacitive divider. The difference is instead of using only two sensing capacitances, four sensing capacitances are connected in a bridge form.

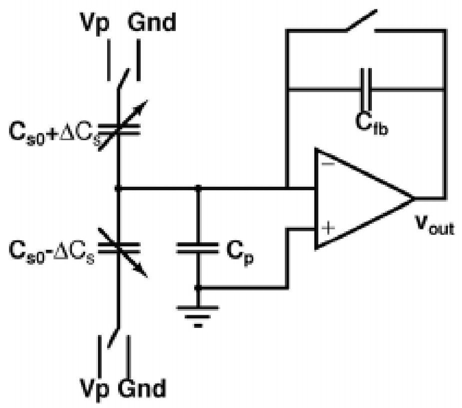


Figure 5 Capacitive Divider Half Bridge (Sun, H. et al., 2011)

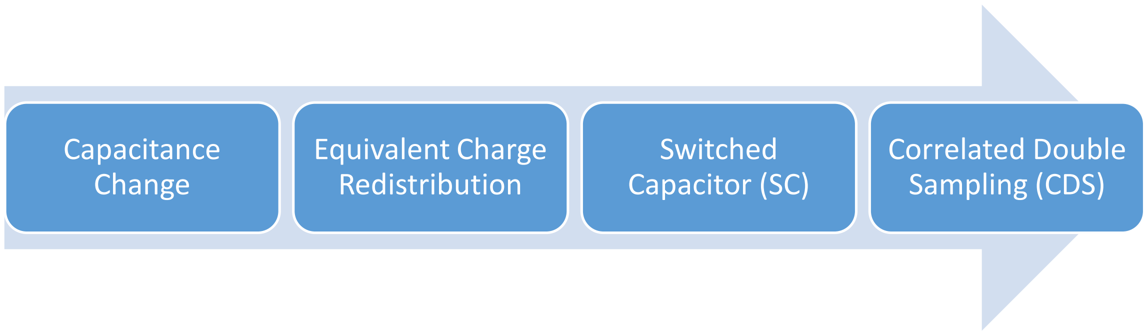

Capacitance Change
Equivalent Charge Redistribution
Switched Capacitor (SC)
Correlated Double Sampling (CDS)

Figure 6 Charge Sensing Readout Process

The process of output readout by using this architecture is shown in Figure 6. The change of capacitance first being detected and later the value will be converted into equivalent charge redistribution. This charge redistribution later being detect by the Switched Capacitor (SC) circuit. Commonly, the architecture will used Correlated Double Sampling (CDS) technique to eliminate the flicker noise (Fang, 2006). Since the circuit contain a feedback, the output signal taken will be in the ratio of the sensed capacitances changes and the feedback capacitance as shown in Equation 1, where V_{out} is the output voltage, V_p is the input voltage, ΔC_S is the sensed capacitance changes, and C_{fb} is the feedback capacitance value.

$$V_{out} = V_p \frac{\Delta C_S}{C_{fb}} \qquad (1)$$

Capacitive sensing architecture is insensitive to undesired changing, leakage currents, as well as stray and parasitic capacitances. Other than that, it is simple to construct and required small silicon area. This leads to easy to design and fabricate the circuit. By combine it with Sigma-Delta modulation technique, the output can be obtained in digital form. However, Charge Sensing is highly affected by offsets, thermal noise, flicker noise, and clock feed-through noise which contribute to less precision and accuracy. To overcome this, Correlated Double Sampling (CDS) is suggested by Yazdi, Kulah, and Najafi (2004) which working by taking difference of the measured signal at the foreground and the background.

2.1.2 Current Sensing

Current Sensing architecture done in continuous time and known as Continuous Time Current (CTC) Sensing. There are two methods in using this architecture, namely Trans-Impedance Amplifiers (TIA) and Trans-Capacitance Amplifiers (TCA), as shown in Figure 7. Basically, the process started with changes in sensing capacitance and this will produce charge transfer. This charge transfer then will be detected by utilizing TIA or TCA. There are several advantages of using this architecture in detecting change in the capacitance value. The main advantages are the ability to achieve high gain and insensitive to parasitic capacitances (Sun, Jia, Liu, Yan, & Hsu, 2011). In addition, CTC can remove dc biasing easily. However, CTC gives a very high flicker noise and white noise.

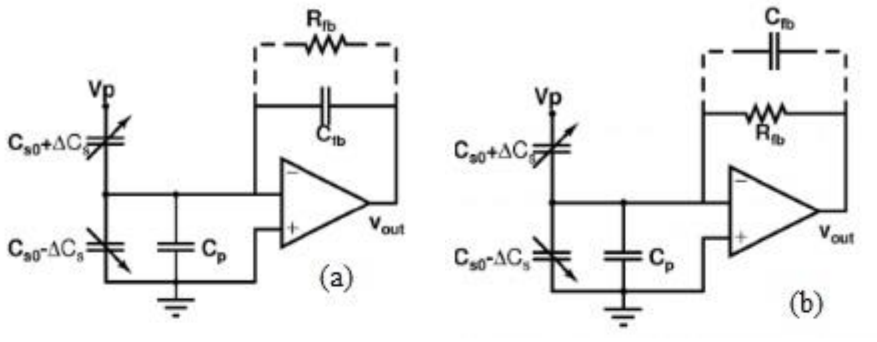


Figure 7 Current Sensing Topology (a) TCA (b) TIA (*Sun, H. et al., 2011*)

Since CTC is using feedback system also, the output voltage also calculated by considering the feedback system. In TCA, since both are in capacitance value, the output voltage can be calculated by taking the ratio between the sensed capacitance changes and feedback capacitances. The only differences than Charge Sensing is the circuit does not used the switch in the feedback system. The equation to find output voltage for TCA can be found in Equation 2, where V_{out} is the output voltage, V_p is the input voltage, ΔC_S is the sensed capacitance changes, and C_{fb} is the feedback capacitance value.

$$V_{out} = V_p \frac{\Delta C_S}{C_{fb}} \qquad (2)$$

While for TIA, the capacitance need to be converted into equivalent resistance value. The output voltage equation for TIA can be found in Equation 3, where V_{out} is the output voltage, V_p is the input voltage, ω is the angular frequency, ΔC_S is the sensed capacitance changes, and R_{fb} is the feedback resistance value.

$$V_{out} = V_p \cdot j\omega \Delta C_S R_{fb} \tag{3}$$

2.1.3 Voltage Sensing

The voltage sensing architecture as shown in Figure 8 is quite simple as it directly amplifying the change of voltage detect in the sensing capacitances. It is considered as Continuous Time Voltage (CTV) Sensing technique as it is operate in continuous time condition. There are two structure offered by this architecture which are Closed Loop and Open Loop. Both have their own advantages and disadvantages. Closed loops offers accurate gains and better linearity but need to trade of with high power consumption as well as high noise. As for open loop, it offers benefits including lower gain, low-power and low-noise.

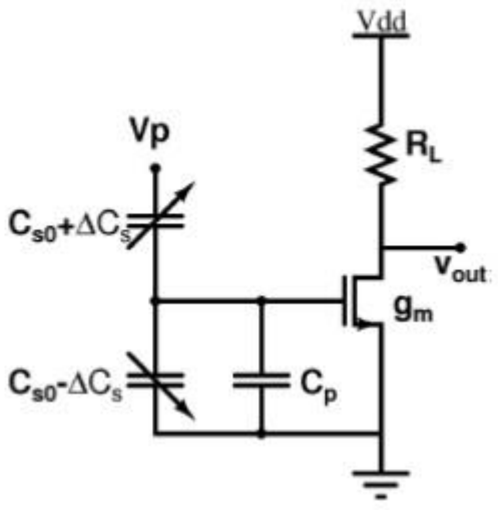


Figure 8 Voltage Sensing Architecture (Sun, H. et al., 2011)

Usually, CTV will be combined with Chopper Stabilization Technique which can be seen in Figure 9.

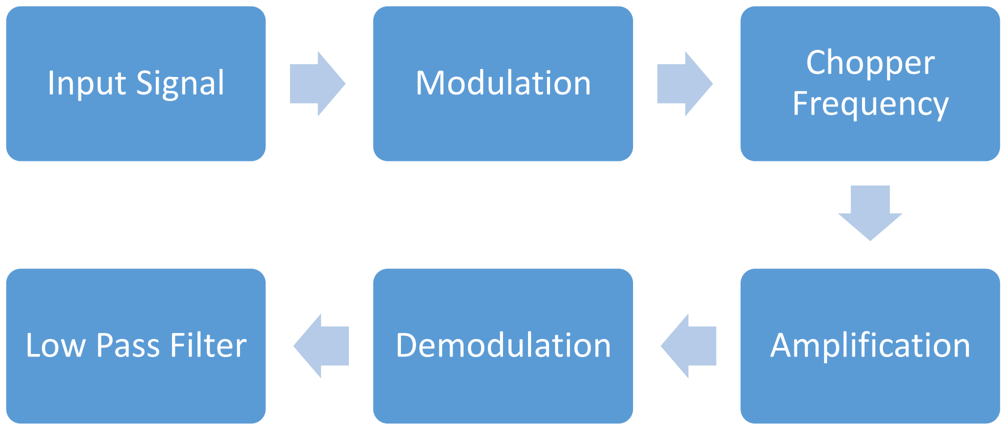


Figure 9 Continuous Time Voltage (CTV) with Chopper Stabilization Technique

The input signal will be sensed first by the circuit. Then, the signal will be modulated to a much higher frequency, called chopper frequency where the amplification will be done. At this stage, the signal is in the odd harmonics of frequency. In addition, the amplification is done at this stage due to small presence of flicker noise here. The signal is then demodulated back to the baseband frequency prior to low pass filter the noise. Use of chopper stabilization technique will minimize the DC offset, Flicker Noise, and associated low frequencies noises (Wu, Fedder, & Carley, 2004).

The output voltage can be calculated by using equation 4, which taken into account the value of input voltage (V_p), Sensed Capacitance Changes (ΔC_S), transconductance of the MOSFET (g_m), Load Resistance (R_L), Initial Sensed Capacitance (C_{S0}), and Parasitic Capacitance (C_P).

$$V_{out} = V_P \cdot \Delta C_S \cdot \frac{g_m R_L}{(2C_{s0}+C_p)} \qquad (4)$$

CTV offers benefits in term of high linearity and stability output, superior noise performance, and reduce switch noise. In addition, the amplification can be improved by adding the differential difference amplifiers.

2.2 NOISE SOURCES IN CAPACITIVE CMOS-MEMS READOUT CIRCUITS

The Noise in CMOS-MEMS readout circuit can be consist of few type of noises that combined together to become a very high noise. The source of noise in the readout circuit is shown in Figure 10.

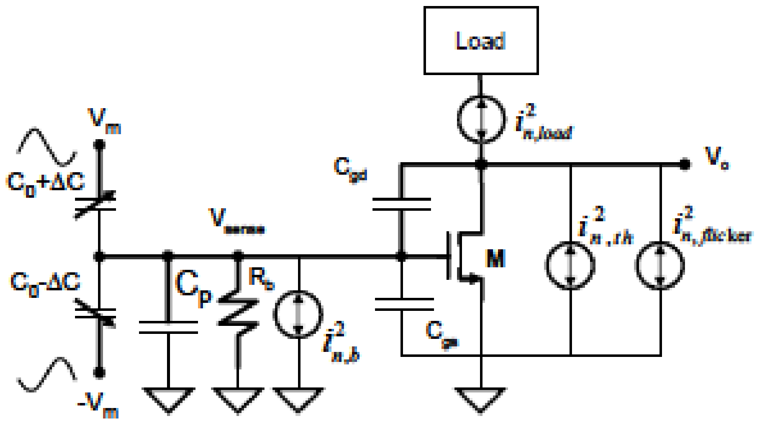


Figure 10 Noise sources of capacitive sensing

C_p is the parasitic capacitance, R_b is the Brownian resistance, $i^2_{n,b}$ is the Brownian noise current, C_{gd} is gate-drain capacitance, C_{gs} is the gate-source capacitance, $i^2_{n,load}$ is the load noise current, $i^2_{n,th}$is the thermal noise current and $i^2_{n,flicker}$is the flicker noise current. All of these noises will be discussed in this section.

2.2.1 Brownian Noise

Brownian Noise comes from random Brownian motion of molecules which leads to electro-mechanical noises. Several method can be implement in order to reduce Brownian Noise including by increasing the mass and decrease the damping force by reducing the height of the sensed capacitor or extending the gap (Tsai and Fedder, 2005).

2.2.2 Electronic Noises

The most significant noise in the CMOS-MEMS circuits is electronic noises. The noises comprised of few type of noises including modulation signal noises, Thermal noises, leakage noises, and Flicker noises (1/f noises) (Wu, Fedder, and Carley, 2004). The flicker noise current can be found based on equation 5.

$$\overline{i_{n,fn}^{2}} = \frac{K_f}{C_{ox}WL}\frac{g_m^2}{f} \qquad (5)$$

As in the equation, $i_{n,fn}$ is the flicker noise current, K_f is the Process-dependent constant, C_{ox} is the Gate Capacitance per unit Area, W is the Width, L is the Length, g_m is the Trans-conductance value, and f is the frequency.

Modulation signal noises generated between modulation voltage and the sensing capacitance. Wu, Fedder, and Carley (2004) further suggested that these noises can be removed by applying different topology at the front end which the generated noises will cancel out each other.

Thermal noises is created due to free carriers' random motion between drain and source channel. Theoretically, as the absolute working temperature increases, the thermal noises also increased. However, the noise does not related to frequency. To overcome this type of noise, a maximum aspect ratio which includes the width and the length need to be applied. Other than that, trans-conductance also need to be kept at maximum. The equation to the thermal noise can be found in equation 6.

$$i_{nthermal}^{2} = 4\gamma TK'(\frac{W}{L})V_{ov} \qquad (6)$$

In equation 6, $i_{nthermal}$ is the thermal current, γ is the MOSFET thermal noise coefficient, T is the temperature, K is the Boltzmann's constant, W is the width, L is the length, and V_{ov} is the overdrive voltage.

On the other hand, shot noise or leakage noise also gives significant effect to the readout circuits. It is suggested that the noises are either the reverse-biased diode noise leakage or the total sum of thermal noise. Shot noise is inversely proportional to square modulation frequency and it makes significant presence in very low frequencies. To cope with this noises, it is recommended to choose modulation frequency with small leakage current. The equation of shot noise can be found as equation 7, where $i_{nleakage}$ is the shot noise current, q is the electron charge, and I_{leak} is the leakage current.

$$i_{nleakage}^{2}(f) = 2qI_{leak} \tag{7}$$

In MOSFETS, flicker noises contributed the highest source of noise in the circuit. Even though the origin of the noises is still debatable, the noises could give a massive problem in readout the output signals from the circuits. Since flicker noise is inversely proportional to active gate area (Width-Length) (Tan et al., 2011) and operating frequencies (f), it is advisable to use maximum value for both parameters in order to achieve high noise performances.

2.2.3 Parasitic Capacitances

Parasitic capacitances also need to be taken into consideration as it will affect the resolution and noise of the overall circuit. The capacitances that can be considered as these capacitances are the parasitic capacitances (C_p), Gate-Source Capacitances (C_{gs}), and Gate-Drain Capacitances (C_{gd}). Both equation for C_{gs} and C_{gd} can be found in equation 8 and 9 respectively.

$$C_{gs} = \frac{2}{3}WLC_{ox} + WL_{ov}C_{ox} \tag{8}$$

$$C_{gd} = WL_{ov}C_{ox} \tag{9}$$

In above equations, C_{gs} is the Gate-Source Capacitance, C_{gd} is the Gate-Drain Capacitance, W is the width, L is the Length, C_{ox} is the Gate Capacitance per unit Area, and L_{ov} is the Overlapping Length.

CHAPTER 3:

METHODOLOGY/ PROJECT WORK

After a careful and full consideration, the Continuous Time Voltage (CTV) sensing architectures has been chosen. The project work is divided into two parts, the first part is mathematical modeling by using MATLAB program while the other part is circuit simulation by using MultiSim software.

By referring to Figure 11, to apply voltage sensing architecture technique, the sensed input signal will be modulated to a chopper frequency which is a frequency that is much higher than the initial frequency. The chopped signal later will be amplified before being demodulated back to the baseband. Lastly, the signal will be undergo Low Pass Filter to remove the remaining noise and offsets at odd harmonics of chopping frequency.

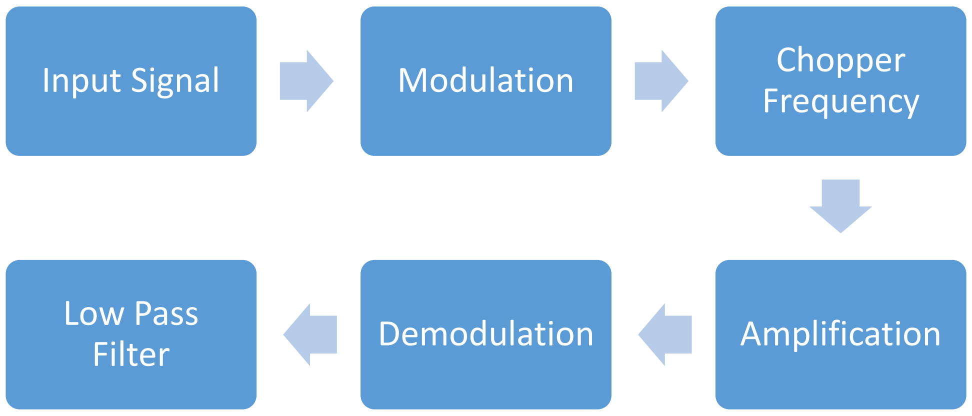


Figure 11 Project Block Diagram

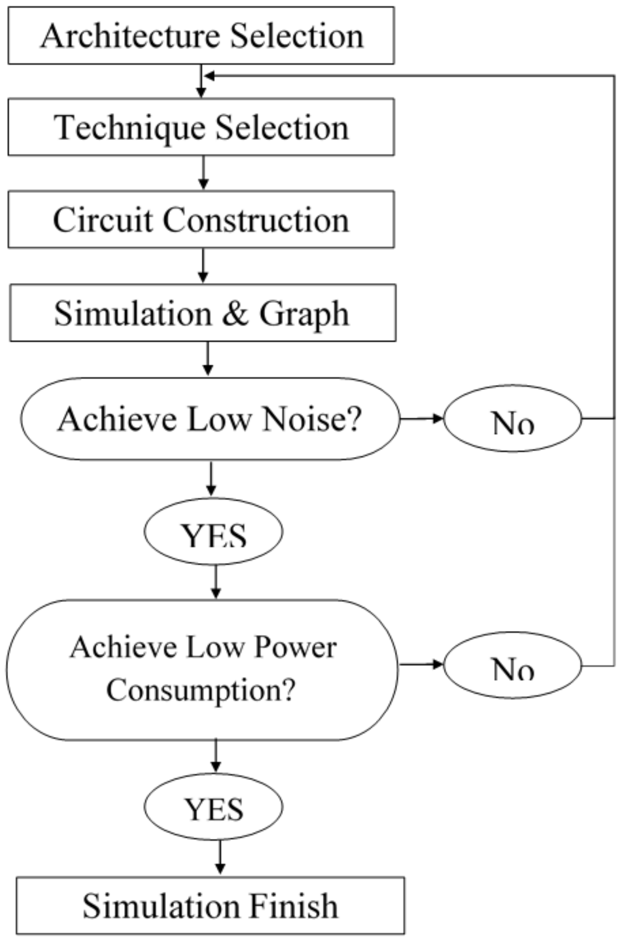


Figure 12 Project Flow Diagram

The flow of the project can be seen in Figure 12. First, an architecture will be selected, which in this case is the Continuous Time Voltage (CTV) architecture. Then, a technique will be selected, which for this project, will use the chopper stabilization technique. Then the circuit will be constructed, both by mathematical modeling in MATLAB and circuit simulation in MultiSim. From the simulation and graph, the output will be evaluated by two criteria, which are first, “does it achieve a low noise?”, and secondly, “Does it achieve a low power consumption”. If not for any of these condition, the circuit will be reevaluate especially on the technique done. If both low noise and low power consumption have been achieved, then the work for this project is done.

Tools & Software

Tools/ Software	**Description**
MATLAB	**Used to simulate mathematical modeling of the project** MATLAB is a good software that can be used for algorithm development, data visualization, and data analysis. Since the project includes various calculation and parameter, MATLAB is the best tool to be used.
MultiSim	**Used to simulate the circuit of the project** MultiSim is a software to develop a circuit without the need of physical part through simulation. Few programs has been considered to do the simulation including PSpice and Cadance, but with simple and user-friendly interface, MultiSim is sufficient for the use of this project.
Microsoft Office	**Used to write report and presentation slides** Microsoft Office is a package of software that offers tools to write reports such as Microsoft Word, Microsoft Power Point, and Microsoft Excel.

Table 1 Tools & Software

Selection of the parameter

The parameter of the project is carefully chosen and can be observed in Table 1.

Input Signal	
Type	Sine
Voltage	1µV(p-p)
Bandwidth	75-105Hz
Frequency	90 Hz
Chopper Signal	
Type	Square
Frequency	16kHz
Voltage	1V(p)
Noise	
Magnitude	150nV
Amplifier	
Gain	50 (34dB)
Bandwidth	32kHz

Table 2 Circuit Parameter

Justification on selection of the parameter

The input signal detected by the sensing devices is expected to be 1µV. This reflects the real scenario of the devices that may detect signal in the range of femto- (10^{-15}) to atto- (10^{-18}) farad. Other than that, the simulation will be done in the interest of bandwidth between 75 and 105 Hz. This is because frequency higher than 105 Hz may having low impact on the SNR while taking frequency lower than 75 Hz is not productive to be evaluate (Abdelkader, 2010). Thus, since the bandwidth is only 30 Hz, taking a sinusoidal signal with 90 Hz of frequency as the input is sufficient enough to evaluate the performance of the circuit.

For the chopper frequency, a square wave signal with 1V amplitude will be used. This will convert the input signal into odd harmonics frequencies prior of amplification stage. The frequency of the square wave is chosen to be 16 kHz so that the flicker (1/f) noise will still small without the need to compensate the transistor size in the core amplifier.

The gain of the amplifier is set to be 50 (34dB) to avoid the common-mode signal will be converted into a differential signal. The bandwidth of the amplifier also is set to be twice of the modulation frequency (32 kHz) to reduce the effect of aliasing of the signal. This is due to Nyquist Theorem that suggested the sampling rate must be at least twice of maximum frequency.

The output of the circuit is expected to achieve the value as shown in Table 2.

Expected Outcome	
SNR (dB)	6 dB
Power Dissipation	110 µW
Size	0.25 mm^2
Allowable Noise	<180nV

Table 3 Expected Output

A signal-to-noise ratio (SNR) of 6 dB is choosen based on other related paper that has achieved above 5 dB of Output SNR. This is achievable and sufficient with the input of 90 Hz sinusoidal. Since a minimum of 6 dB output is expected from the circuit, the total tolerable noise can be computed by using Equation 10.

$$\mathrm{A_{rms}} = \frac{1}{2} \cdot \frac{1}{\sqrt{2}} \cdot \mathrm{A_p} \tag{10}$$

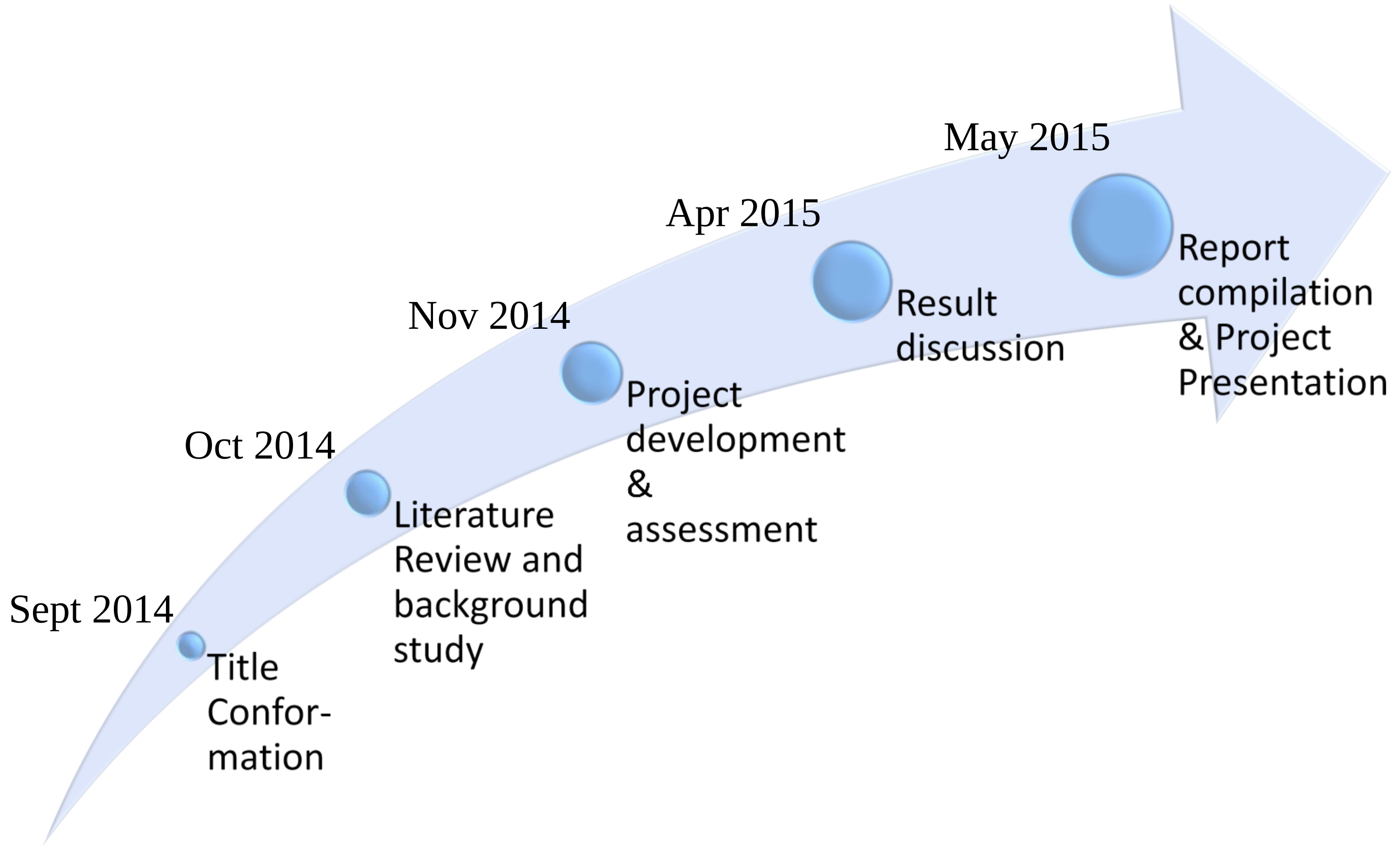
Where A_{rms} is the amplitude of the input signal in root-mean-square (rms) value and A_p is the peak amplitude of the input signal. In this case, the amplitude is 1μV. A_{rms} is calculated and producing 3.536×10^{-7} V. The total allowable noise is later computed by using SNR_{dB} formula in Equation 11.

$$SNR_{dB} = 20 \cdot log_{10}\left(\frac{A_{rms}}{N_{tot}}\right) \tag{11}$$

Where SNR_{dB} is 6dB and N_{tot} is the total noise. From the calculation it is found that the the total noise should not exceed 180 nV.

While the power dissipation and size is not a main concern in this paper, the parameter is still needed to have a high performance circuit.

Key Project Milestone


Sept 2014
Title Confor-mation
Oct 2014
Literature Review and background study
Nov 2014
Project development & assessment
Apr 2015
Result discussion
May 2015
Report compilation & Project Presentation

Project Timeline Study Plan (FYP I)

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project Topic Selection	🟪	🟪	🟪											
Research on studies & projects				🟪	🟪	🟪								
Submission of Extended Proposal						▨								
Software Training						🟪	🟪							
Initial Project Development							🟪	🟪						
Proposal Defense								▨						
Further Project Development									🟪	🟪	🟪	🟪	🟪	🟪
Result Evaluation										🟪	🟪	🟪	🟪	🟪
Submission of Interim Draft Report													▨	
Submission of Interim Report														▨

Project Timeline Study Plan (FYP II)

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project Work Continues	■	■	■	■	■	■	■	■	■	■	■	■		
Submission of Progress Report							■							
Pre-SEDEX										■				
Submission of Draft Final Report											■			
Submission of Dissertation (Soft Bound)												■		
Submission of Technical Paper												■		
Viva													■	
Submission of Project Dissertation (Hard Bound)														■

3.1 MATLAB

The work for this part has been divided into 13 separately parts that later will work together to work as one mathematical modeling. All the coding used can be refer in the appendices section. The parts are as follow:

1. Main coding

   This is the main part of the coding where all the parameter is defined and all the functions are included in here.
2. Sine-Wave Generator

   Coding to generate a sine-wave input to the system. This sine-wave will act as an original signal before anything is done including modulation, amplification, and insertion of noise.
3. Square-Wave Generator

   Square-wave is required to modulate the signal into chopper frequency and then after the amplification, the square wave is once again used to demodulate it back to the baseband.
4. High Pass

   High Pass function will filter the signal so that the lower frequency corner of the amplifier is 1 Hz.
5. Low Pass

   Low pass filter is used to remove the noise presented in the signal. This is done after the signal is demodulated back into the baseband.
6. Noise-Generation

   This is a coding to generate noise. In this work, the noise includes Flicker noise (1/f noise) and thermal noise.
7. Anti-Aliasing

   As in other signal processing, aliasing problem might occurred. To counter this problem in this work, Anti-Aliasing Function (AAF) is used.
8. Band Pass

   To filter out the input signal to a specific order, this function is used.

9. FFT Magnitude

   The function will compute the magnitude of the fast Fourier transform (FFT) of the output signal.

10. Hodie Window

    The Hoodie Window function is used to minimize the spectral blurring of the output signal.

11. SNR Computation

    This function is used to calculate the value of Signal-to-Noise Ratio (SNR) of the output signal. The performance of the signal is evaluated here.

12. Time Domain Plot

    This function is used to plot the output of the signal in time domain.

13. Frequency Domain Plot

    This function is used to plot the output signal in the frequency domain.

3.2 MultiSim

The circuit is design based on Fang (2006) paper but with certain alteration and the overall circuit. A basic circuit without the chopper frequency can be seen in Figure 13.

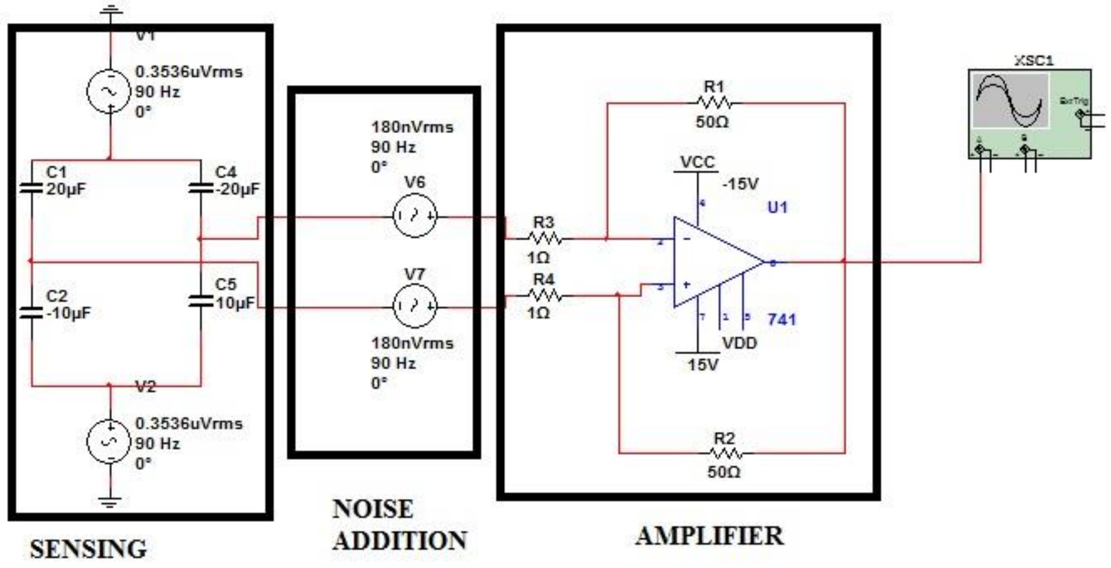


Figure 13 Circuit without Chopper Diagram

A sine wave with rms value of 0.3536 µV and frequency of 90 Hz is use as the input signal. Then, the noise is added after the sensing circuit. The noise use a sine wave with rms value of 180 nV and frequency of 90 Hz. Lastly, the signal will go through an amplification circuit with a gain of 50. Then the output signal is obtained and evaluated.

The improvement circuit that take into account the chopper stabilization technique can be seen in Figure 14.

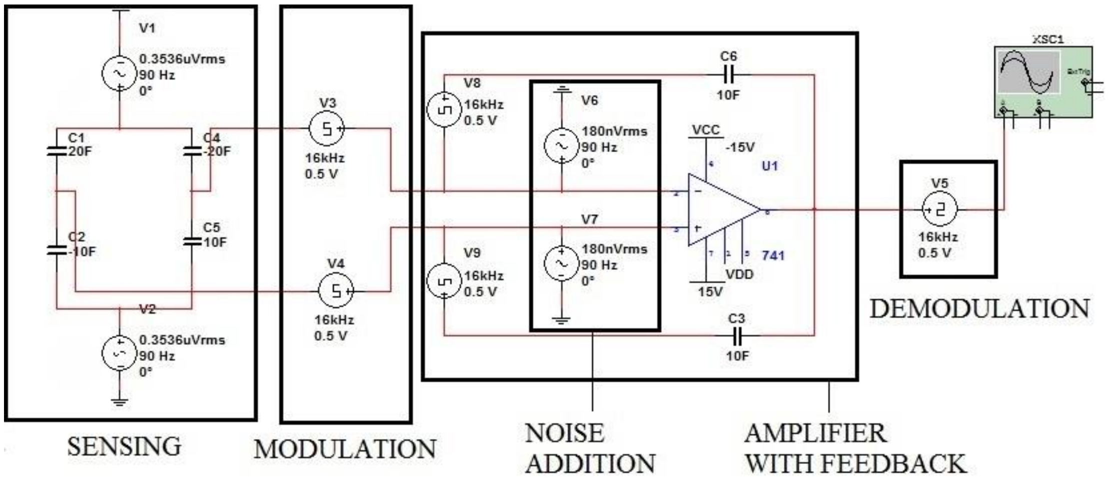


Figure 14 Overall Circuit with Chopper Diagram

The sensing circuit used is just the same, which has sine wave with rms value of 0.3536 μV and frequency of 90 Hz. Then the signal will be transformed into odd harmonic of frequencies by the square wave with amplitude of 0.5 V (peak) and frequency of 16 kHz. Then, the noise of sine wave with rms value of 180 nV and frequency of 90 Hz is added into the signal. The output from this noise addition stage will then be amplified prior to demodulate back the signal back to the baseband. The amplification stage has a feedback system to improve the performance of the circuit itself.

To calculate the SNR$_{dB}$ of the output, we need a noise value from the circuit. Thus, a noise circuit as shown in Figure 15 is constructed.

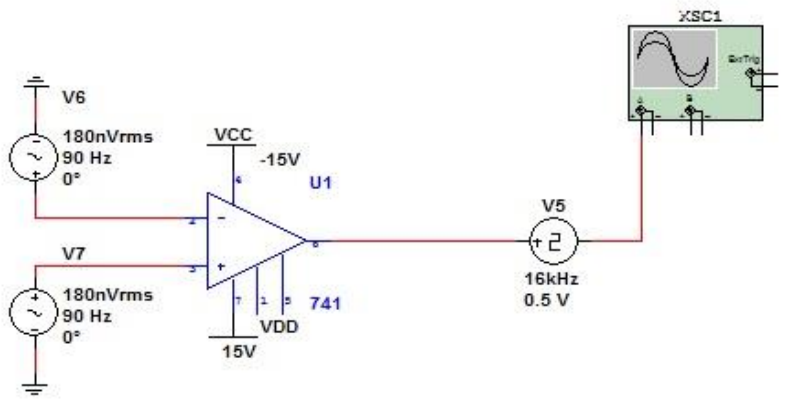


Figure 15 Noise Circuit Diagram

Basically, the input of the noise is just the same as two previous circuit (sine wave with rms value of 180 nV and frequency of 90 Hz). Then the amplification is done and the output of the circuit can be evaluated. Based on the results obtained from the three circuits, the SNR$_{dB}$ value is calculated based on the Equation 11.

$$SNR_{dB} = 20 \cdot log_{10}\left(\frac{A_{rms}}{N_{tot}}\right) \quad (11)$$

3.3 Calculation

1. Sensing Circuit

   To obtain the value of the output voltage, the reactance of the capacitance is first obtained, as shown in Equation 12.

$$X_{Cn} = \frac{1}{2\pi f C_n} \tag{12}$$

   After each of the reactance value is obtained, the total reactance is calculated by taking the sum of each reactance as shown in Equation 13.

$$X_{C(TOTAL)} = \sum X_{Cn} \tag{13}$$

   The source voltage will be divided by the total reactance to obtain the current in the circuit, as shown in Equation 14.

$$I = \frac{V}{X_{C(TOTAL)}} \tag{14}$$

   Finally the voltage across each capacitance is computed by using equation 15.

$$V_n = I \cdot X_{Cn} \tag{15}$$

2. Modulation Circuit

   Basically the output from the modulation circuit will be obtained by using equation 16.

$$V_{mod} = V_{in} \times A_{mod} \tag{16}$$

3. Noise Addition

   The noise source is inserted by arranging it in series with the output from the modulation circuit. Thus, the output will be computed by equation 17.

$$V_{OutNoise} = V_{mod} \times V_{noise} \tag{17}$$

4. Amplification

After the noise is added, the amplification is done. The output can be calculated by using equation 18.

$$V_{amp} = V_{OutNoise} \times A_{amp} \tag{18}$$

5. Demodulation Circuit

The output of the amplification stage is then modulated back to baseband, can be calculated by using equation 19.

$$V_{demod} = V_{amp} \times A_{demod} \tag{19}$$

CHAPTER 4:

RESULTS AND DISCUSSION

4.1 MATLAB

MATLAB tool has been used to simulate the mathematical modeling of the work. For this work, works by Abdelkader H. S. entitled "Design of a Chopper Amplifier for Use in Biomedical Signal Acquisition" has been used as reference. The result is shown in Table 4, Figure 11, and Figure 12.

Circuit	SNR_{dB}
With Chopper Configuration	-35.39
Without Chopper Configuration	-33.76

Table 4 MATLAB Output

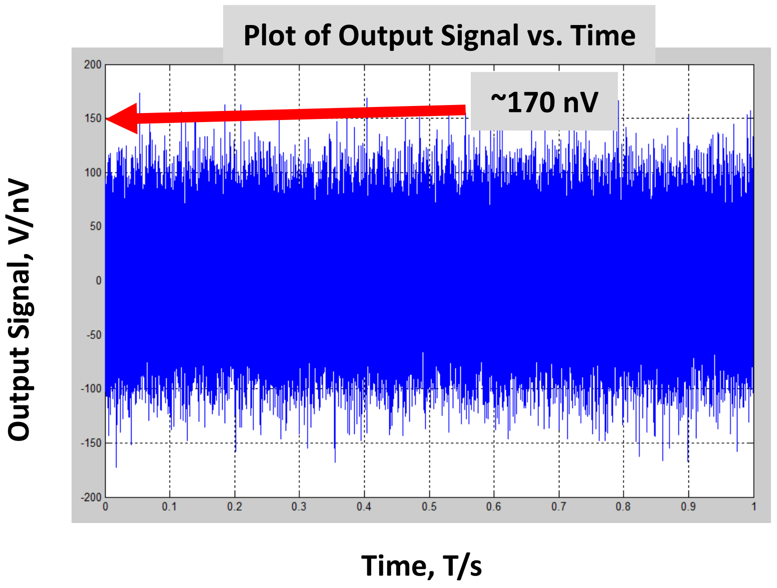


Figure 16 Output With Chopper Configuration

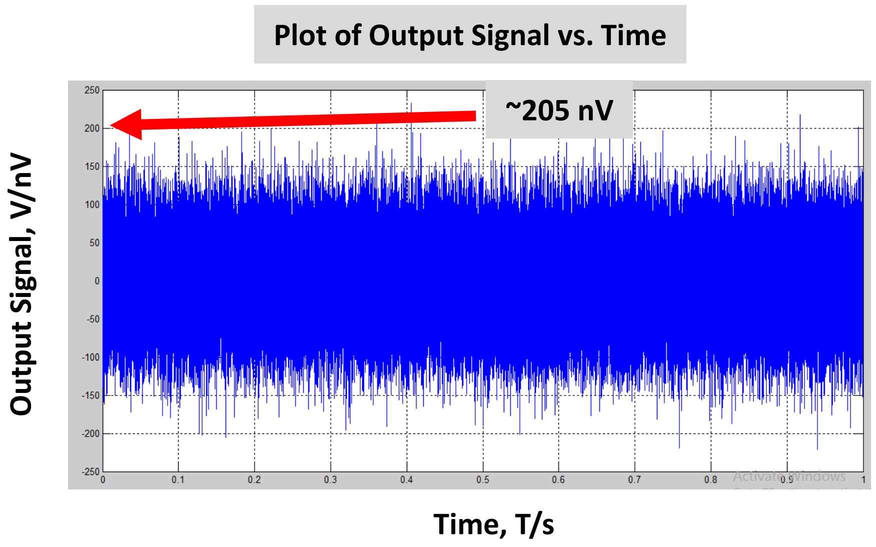


Figure 17 Output without Chopper Configuration

The noise generation output also has been obtained, both in Time Domain and Frequency Domain, as can be seen in Figure 18 and 19 respectively.

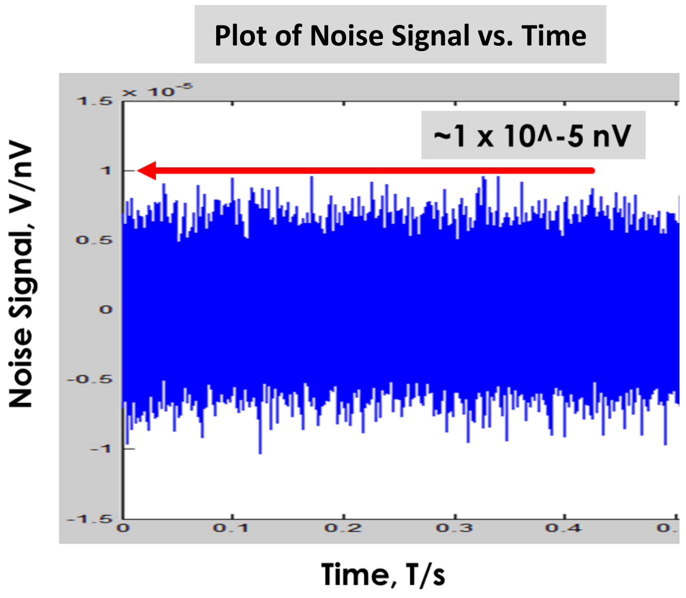


Figure 18 Noise Output in Time Domain

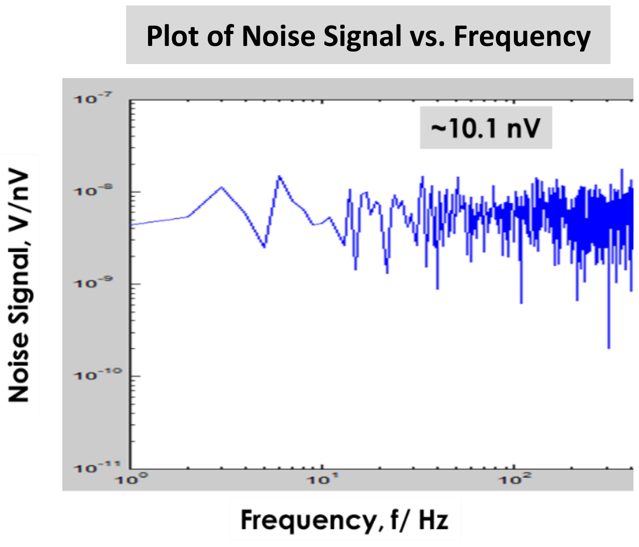


Figure 19 Noise Output in Frequency Domain

The SNR of the signal is then computed by using Equation 11.

<u>With Chopper</u>

$$SNR_{dB} = 20 \cdot \log_{10}\left(\frac{A_{rms}}{N_{tot}}\right)$$

$$SNR_{dB} = 20 \cdot \log_{10}\left(\frac{170\ nV}{10^{-5}V}\right)$$

$$SNR_{dB} = -35.39\ dB$$

<u>Without Chopper</u>

$$SNR_{dB} = 20 \cdot \log_{10}\left(\frac{205\ nV}{10^{-5}V}\right)$$

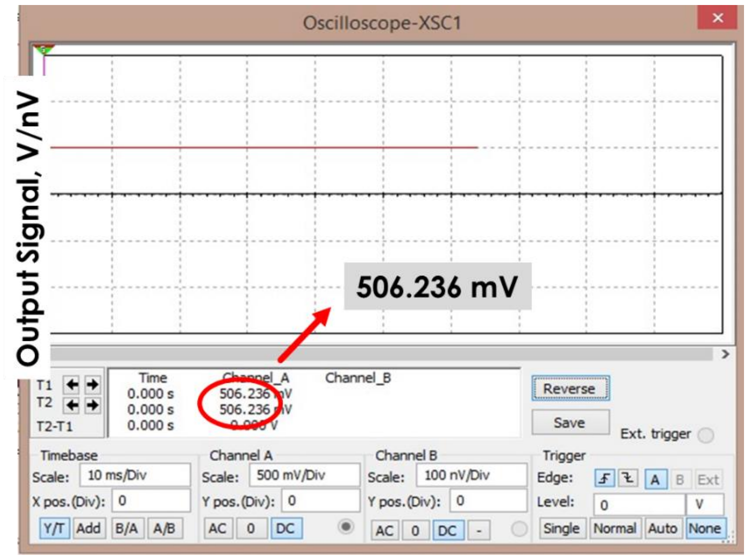
$$SNR_{dB} = -33.76\ dB$$

4.2 MultiSim

Four circuits has been simulated, which are the categorized as first, Circuit with Chopper Configuration and the other is Circuit without Chopper Configuration. For each type, two circuit is simulated, basically to ease in the calculation of the SNR_{dB}, which are the complete circuit and the noise circuit.

<u>Circuit without Chopper</u>

The circuit in Figure 13 is simulated and producing the result as shown in Figure 22.

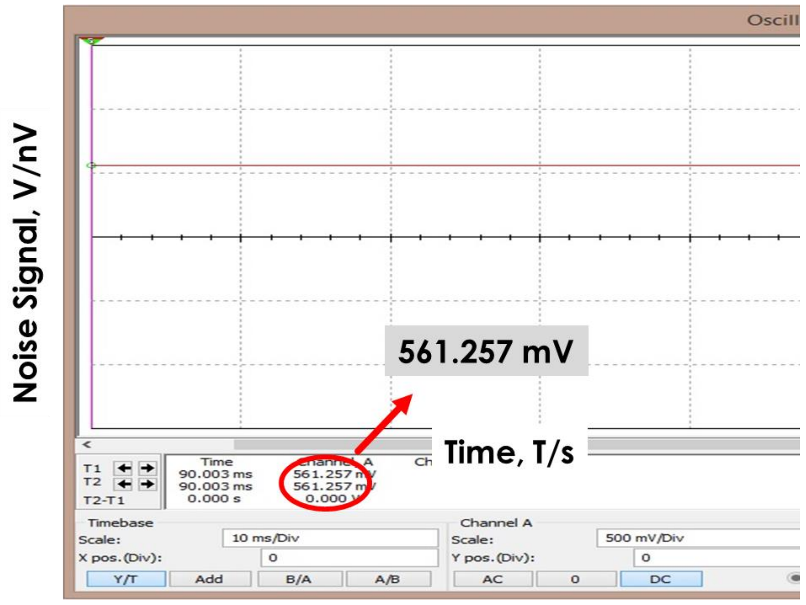


```
      Time      Channel_A     Channel_B
T1    0.000 s   506.236 nV
T2    0.000 s   506.236 nV
T2-T1 0.000 s   0.000 V
```

Timebase — Scale: 10 ms/Div; X pos.(Div): 0; Y/T
Channel A — Scale: 500 mV/Div; Y pos.(Div): 0; DC
Channel B — Scale: 100 nV/Div; Y pos.(Div): 0; DC
Trigger — Edge: A; Level: 0 V; None

Figure 20 MultiSim Output (Non-Chopper)

From the result above, it is observed that the circuit producing 506.236 nV of output. Later, the noise circuit for non-chopper circuit also being simulated and producing the result shown in Figure 23.



	Time	Channel A
T1	90.003 ms	561.257 mV
T2	90.003 ms	561.257 mV
T2-T1	0.000 s	0.000 V

Timebase Scale: 10 ms/Div; X pos.(Div): 0; Y/T
Channel A Scale: 500 mV/Div; Y pos.(Div): 0; DC

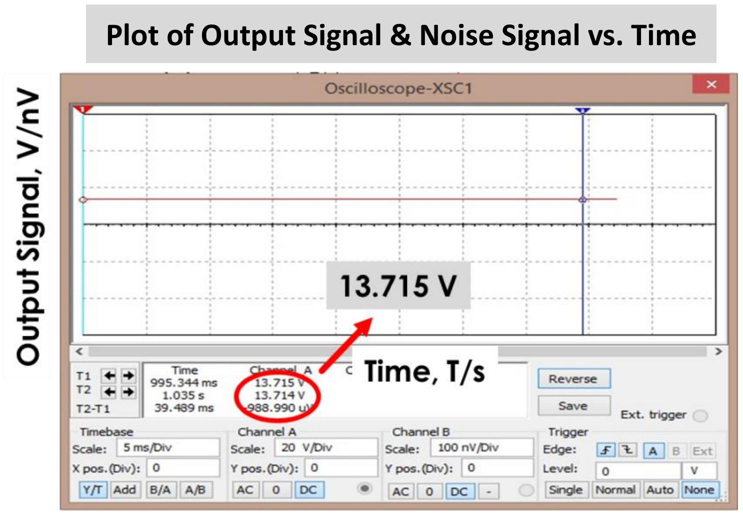
Figure 21 Noise Output (Non-Chopper)

The SNR value of the circuit is then calculated by using equation 11.

$$SNR_{dB} = 20 \cdot \log_{10}\left(\frac{506.236\, mV}{561.257\, mV}\right)$$

<u>Circuit with Chopper</u>

The circuit in Figure 14 is simulated and the output of the circuit is shown in Figure 20.



	Time	Channel A
T1	995.344 ms	13.715 V
T2	1.035 s	13.714 V
T2-T1	39.489 ms	-988.990 uV

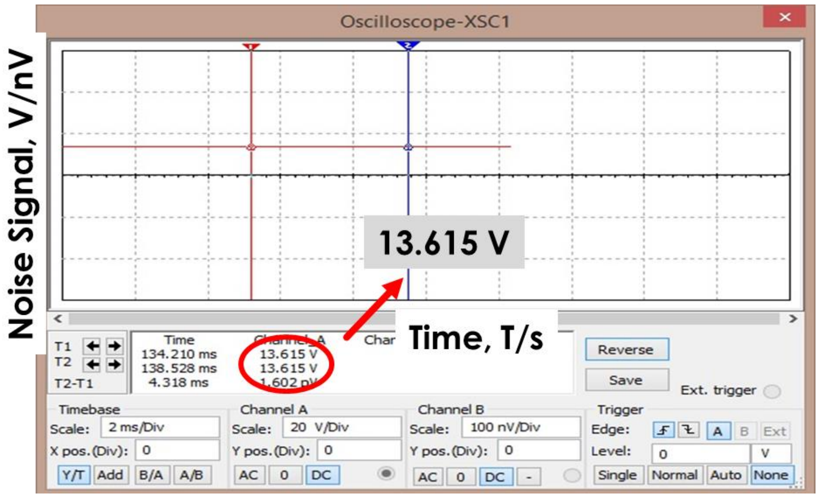
Timebase Scale: 5 ms/Div; X pos.(Div): 0; Y/T
Channel A Scale: 20 V/Div; Y pos.(Div): 0; DC
Channel B Scale: 100 nV/Div; Y pos.(Div): 0; DC
Trigger Edge: A; Level: 0 V; None

Figure 22 MultiSim Output Diagram (Chopper)

As observed, it produces output of 13.715 V.

<u>Noise Circuit</u>

While the circuit for Noise is also constructed and simulated. The circuit producing the result in Figure 21.



	Time	Channel A
T1	134.210 ms	13.615 V
T2	138.528 ms	13.615 V
T2-T1	4.318 ms	1.602 pV

Timebase Scale: 2 ms/Div; X pos.(Div): 0
Channel A Scale: 20 V/Div; Y pos.(Div): 0
Channel B Scale: 100 nV/Div; Y pos.(Div): 0
Trigger Level: 0 V

Figure 23 MultiSim Noise Output (Chopper)

As observed in diagram above, it producing output of 13.615 V.

To calculate the SNR$_{dB}$, Equation 11 is used.

$$SNR_{dB} = 20 \cdot \log_{10}\left(\frac{13.715\ V}{13.615\ V}\right)$$

$$= 63.563 \times 10^{-3} dB$$

$$= -0.\ 896 dB$$

CHAPTER 5:

CONCLUSION AND RECOMMENDATION

As the conclusion, it is observed that from the mathematical modeling in MATLAB and circuit simulation of MultiSim, the circuit with Chopper Configuration show a better result compared without the Chopper Configuration. However, the Signal-to-Noise Ratio (SNR) achieved is not as expected, which is much lower. This might be because the circuit is not properly constructed and more improvement can be done in the future. This includes a proper selection of the parameter especially for the capacitance value that play major role in determining the gain of the circuit. Other than that, the simulation used a non-variable capacitance which is not simulated the real circuit due to restriction of the software used. Thus, the value obtained might not be accurate but it does simulate the concept used in the Capacitive Sensing. In future work, more advanced software is suggested to simulate the real situation.

REFERENCES

An Introduction to MEMS, 2002, Loughborough, LE11: PRIME Faraday Partnership

Abdelkader Hadj Said, 2010, "Design of a Chopper Amplifier for Use in Biomedical Signal Acquisition", Unpublished master's thesis, Southern Illinois University Edwardsville, Edwardsville, Illinois

Fang, D., and Xie, H., 2006, "A Low-Noise Low-Power Preamplifier for Capacitive CMOS-MEMS Gyroscopes", *Proceedings of the 2006 49th IEEE International Midwest Symposium on Circuits and Systems* **2:** 270-274

Fang, D., 2006, "Low-Noise and Low-Power Interface Circuits Design for Integrated CMOS-MEMS Inertial Sensors", Unpublished doctor of philosophy's thesis, University of Florida, Gainesville, Florida

Navid Yazdi, Haluk Kulah, and Khalil Najafi, 2004, "Precision Readout Circuits for Capacitive Microaccelerometers", *Proceedings of IEEE Sensors 2004* pp. 28-31

Shekhar Borkar, 2006, "Electronics Beyond Nano-scale CMOS", *Proceedings of the 43rd Annual Design Automation Conferenc*e pp. 807-808

Sun, H., Jia, K., Liu, X., Yan, G., Hsu, Y.-W., and Xie, H., 2010, "A Low Temperature-Dependence Gain-Boosting Front-Eng Amplifier for CMOS-MEMS Gyroscopes", *2010 IEEE Sensors* pp. 1029-1032

Sun, H. et al., 2011, "A CMOS-MEMS Gyroscope Interface Circuit Design with High Gain and Low Temperature Dependence", *IEEE Sensors Journal* **11(11):** 2740-2748

Tan, S.-S., Liu, C.-Y., Yeh, L.-K., Chiu, Y.-H., Lu, M. S.-C., and Hsu, K.Y.J., 2011, "An Integrated Low-Noise Sensing Circuit With Efficient Bias Stabilization for CMOS

MEMS Capacitive Accelerometers", *IEEE Transactions on Circuits and Systems-I: Regular Papers* **58 (11):** 2661-2672

Tsai, J.M., and Fedder, G.K., 2005, "Mechanical Noise-Limited CMOS-MEMS Accelerometer", *Technical Digest of the 18th IEEE International Conference on Micro Electro Mechanical Systems (MEMS 2005)* pp. 630-633

Varadan, V.K., Vinoy, K.J., Gopalakrishnan. S., 2006, Smart Material Systems and MEMS: Design and Development Methodologies. Chincester, UK: John Wiley & Sons.

Witvrouw, A., 2006, "CMOS-MEMS Integration: Why, How and What?", *Proceedings of 2006 EEE/ACM International Conference on Computer-Aided Design* pp.826-827

Wu, J., Fedder, G.K., and Carley L.R., 2004 "A Low-Noise Low-Offset Capacitive Sensing Amplifier for a 50-µg/√Hz Monolithic CMOS MEMS Accelerometer", *IEEE Journal of Solid-State Circuits* **39 (5):** 722-730

Xie, H., and Fedder, K., 2002, "Vertical comb-finger capacitive actuation and sensing for CMOS-MEMS" in French, P. J. *Sensors and Actuators A: Physical*, Delft; Elsevier

APPENDIX A

MATLAB CODE

The Main Code (ChopperTool)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is a tool to analyze the performance of a Chopper Amplifier.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TRUE = 1 ;
FALSE = 0 ;
ReadFromFile = FALSE ; % Set equal to TRUE if you want to import data vector
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 16*131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence
N = 2 ^ (round(log2(N))) ; % 2 to some power
A = 0.5e-6; % Input sine wave amplitude
F0 = 90; % Frequency of input sine wave
BW = 30; % Desired Amplifier bandwidth in Hertz
Fch = 16e3; % Chopping frequency
Asq = 1; % Amplitude of square wave
G = 50; % Amplifier Gain
n1 = 1;
f1 = 1; % high-pass filter: (f1 = 1/2*pi*(R*C2) = 1Hz
w1 = 2*pi*f1;
n2 = 1; % 1st order LPF for a limited GBW of the op-amplifier
f2 = 2*Fch; % cut-off frequency for a limited GBW (fc0=2*Fch = 32KHz)
w2 = 2*pi*f2;
Rnp = 3.071e4; % thermal noise Resistance of input devices
Kfp = 4.028e-13; % flicker noise Coefficient of input devices
Rnn = 5.607e3; % thermal noise Resistance of load devices
Kfn = 8.353e-13; % flicker noise Coefficient of load devices
n3 = 2; % 2nd Order LPF filter at the output of the demodulator
f3 = 12e3; % the cut-off frequency
w3 = 2*pi*f3;
n4 = 4; % 8th Order BPF filter
w4 = [2*pi*75 2*pi*105]; % passband range: 75Hz - 105Hz
t = 1 / Fs ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Error checks
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ErrorCheck(Fs, N, F0) ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Apply sinewave to the input of the Chopper Amplifer
% the Chopper Amplifier consists of a modulator followed by an
% amplification stage, a demodulator, a low pass filter, and finally a band-pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (ReadFromFile == TRUE)
rfid = fopen('Chopper_data_recorder.txt') ;
ChopperOutput = fscanf(rfid,'%g');
L = length(ChopperOutput) ;
if (L < N)
error('ChopperOutput time sequence is too short') ;
```

```
else
ChopperOutput = ChopperOutput(1 : N)' ;
end
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input stage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SineWave = SineWaveGenerator(N, A, Fs, F0); %input signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modulation stage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
squarewave = SquareWaveGenerator(N, Asq, Fs, Fch);
Modulatorout = SineWave.*squarewave;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Amplification stage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% High pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%hp_filterout = HighPass(n1, w1, Modulatorout,t); % apply the filter to an input signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adding noise at the input
%%%%%%%%%%%%%%%%%%%%%%%%%%%
p_noise = NoiseGenerator(Rnp, Kfp, Fs, N); % Noise generation for input devices
n_noise = NoiseGenerator(Rnn, Kfn, Fs, N); % Noise generation for load devices
noise = p_noise + n_noise; % total noise
Amplinput = Modulatorout + transpose(noise); % noise added at the input of the amplifier
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pure amplification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Amp = G*Amplinput;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% low-pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ampliout = LowPass(n2, w2,Amp,t); % apply the low-pass filter to an input signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demodulation stage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
squarewave = SquareWaveGenerator(N, Asq, Fs, Fch);
Demodulatorout = Ampliout.*transpose(squarewave);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ouptut of the Chopper Amplifier
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ChopperOutput = Demodulatorout;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the filtring stage (2nd order LPF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Filterout = AAF(n3, w3, ChopperOutput, t); % apply the filter to an input signal
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Signal filtring stage (8th order BPF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BP_Filterout = BandPass(n4,w4,Filterout,t); % apply the filter to an input signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lets use the middle group of samples in each array so that we don't have transients to
% contend with.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%L = length(ChopperOutput) ;
%ChopperOutput = ChopperOutput(L/4+1 : 5*L/4) ;
%L = length(Filterout) ;
%Filterout = Filterout(L/4+1 : 3*L/4) ;
%L = length(DSP_Filterout) ;
%DSP_Filterout = DSP_Filterout(L/4+1 : 3*L/4) ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Take FFTs of the output waveforms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ChopperOutputFFT = FFTmagnitude(transpose(ChopperOutput));
FilteroutFFT = FFTmagnitude(transpose(Filterout));
BP_FilteroutFFT = FFTmagnitude(transpose(BP_Filterout));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the SNR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ChopperOutputSNR = ComputeSNR(ChopperOutputFFT, Fs, F0, BW)
FilteroutSNR = ComputeSNR(FilteroutFFT, Fs, F0, BW)
BP_FilteroutSNR = ComputeSNR(BP_FilteroutFFT, Fs, F0, BW)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot in time domain so we can see of waveform looks reasonable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1) ;
hold on ;
TimeDomainPlot(SineWave, 1, Fs, 'b', 1, 4) ;
TimeDomainPlot(ChopperOutput, 1, Fs, 'g', 2, 4) ;
TimeDomainPlot(Filterout, 1, Fs, 'r', 3, 4) ;
TimeDomainPlot(BP_Filterout, 1, Fs, 'b', 4, 4) ;
hold off ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot in the frequency domain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2) ;
hold on ;
FreqDomainPlot(ChopperOutputFFT, 1, Fs, 'b', 1, 3) ;
FreqDomainPlot(FilteroutFFT, 1, Fs, 'g', 2, 3) ;
FreqDomainPlot(BP_FilteroutFFT, 1, Fs, 'g', 3, 3) ;
hold off ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

<u>Sine-wave Generation function</u>

```
function[SineWave] = SineWaveGenerator(N, A1, Fs, Fo)
%
% Generates a sinusoidal time sequency
%
% N is the number of samples to in the time sequence
% A1 is the peak amplitude of the sine wave
% Fs is the sampling frequency
% Fo is the frequency of the sine wave in Hertz
%
% Fhat is the normalized frequency i.e. F0/Fs
%
%Resolution = 1 ; % Frequency resolution in Hertz
%Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
%N = Fs / Resolution ; % Length of input time sequence (2 to some power)
%N = 2 ^ (round(log2(N))) ;
%A1 = 1e-6; % Input sine wave amplitude
%Fo = 45; % Frequency of input sine wave
Fhat = Fo / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SineWave = A1 * sin(K * index);
%Ts = 1 / Fs ;
%time = Ts * index ;
%plot(time, SineWave)

end
```

Square-wave Generation function

```
function[SquareWave] = SquareWaveGenerator(N, A2, Fs, Fch)
%
% Generates a square wave sequency
%
% N is the number of samples to in the time sequence
% A2 is the peak amplitude of the square wave
% Fs is the sampling frequency
% Fch is the frequency of the square wave in Hertz (chopper frequency)
%
%Resolution = 1 ; % Frequency resolution in Hertz
%Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
%N = Fs / Resolution ; % Length of input time sequence (2 to some power)
%N = 2 ^ (round(log2(N))) ;
%Fch=16e3;
%A2=1;
%
% Fhat is the normalized frequency i.e. Fch/Fs
%
Fhat = Fch / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SquareWave = A2 * square(K * index);
%Ts = 1 / Fs ;
%time = Ts * index ;
%plot(time, SquareWave);
%
end
```

High-pass function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analog high-pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out] = High(n, w, input, t)
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence
N = 2 ^ (round(log2(N))) ; % 2 to some power
n = 1;
f1 = 1; % high-pass filter: (f1 = 1/2*pi*(R*C2) = 1Hz
w = 2*pi*f1;
T = 1 / Fs ;
t = T * linspace(0, N-1, N) ;
[z, p, k] = butter(n,w, 'high', 's'); % Zero-Pole-Gain design
hp = zpkdata(z,p,k); % creates a continuous-time zero-pole-gain model
tfhp = tf(hp); % convert to transfer function model
% plot of the transfer function
h = bodeplot(tfhp); setoptions(h,'FreqUnits','Hz');
setoptions(h1,'FreqUnits','Hz','PhaseVisible','off');
out = lsim(tfhp,input,t); % apply the filter to an input signal
```

Low-pass function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analog low-pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out] = LowPass(n, w, input, t)
%Resolution = 1 ; % Frequency resolution in Hertz
%Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
%N = Fs / Resolution ; % Length of input time sequence
%N = 2 ^ (round(log2(N))) ; % 2 to some power
%Fch = 16e3; % Chopping frequency
%n = 1; % 1st order LPF for a limited GBW of the op-amplifier
%f2 = 2*Fch; % cut-off frequency for a limited GBW (fc0=2*Fch=32KHz)
%w = 2*pi*f2; % the angular frequency
%T = 1 / Fs ;
%t = T * linspace(0, N-1, N) ;
[z, p, k]=butter(n,w, 's'); % Zero-Pole-Gain design of the LPF
lp = zpk(z,p,k); % creates a continuous-time zero-pole-gain model
tflp = tf(lp); % convert to transfer function model
% plot of the transfer function
%h = bodeplot(tflp); setoptions(h,'FreqUnits','Hz');
%setoptions(h1,'FreqUnits','Hz','PhaseVisible','off');
out = lsim(tflp,input,t); % apply the filter to an input signal
```

Noise Generation function

```
function[noise] = NoiseGenerator(Rn, Kf, Fs, N)
%
% Script to generate noise vector containing both white and flicker
% noise components.
%
% Paramters
%
TRUE = 1 ;
FALSE = 0 ;
Ut = 26e-3; % Thermal voltage
qe = 1.602e-19; % Charge of electron
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence (2 to some power)
N = 2 ^ (round(log2(N))) ;
NBW = Fs/2 ; % Desired noise bandwidth
WRITE_FILE = TRUE ;
Rnp = 3.071e4; % Rnp, Resistance for thermal noise generation for input devices
Kfp = 4.028e-13; % Kfp, Coefficient for 1/f noise generation for input devices
Rn = 5.607e3; % Resistance for thermal noise generation for load devices
Kf = 8.353e-13; % Coefficient for 1/f noise generation for load devices
%
% Thermal noise standard deviation
%
sigma = sqrt(4 * Ut * qe * Rn * NBW) ;
sigma_Kf = sqrt(4 * Ut * qe * Kf * NBW) ;
%
% Determine the length of vector in samples
%
T = 1 / Fs ;
%
% Compute time vector
%
t = T * linspace(0, N-1, N) ;
%
% Compute white noise vector
%
wn = sigma * randn(1, N) ;
%
% Compute flicker noise vector
%
fn = randn(1, N) ;
%
% Add the two together
%
noise = wn + fn ;
%
% Plot in time domain so we can see of waveform looks reasonable
%
figure(1) ;
plot(t, noise) ;
%
% And also plot in frequency domain
%
```

```
mag = abs(fft(noise)) ;
DeltaFreq = Fs / N ;
freq = DeltaFreq * linspace(0, N/2-1, N/2) ;
mag = 1/N * mag(1 : N/2) ;
figure(2) ;
loglog(freq, mag) ;
%
% Open a ASCII text file and write results to file
% We need to create a PWL file for Spectre
%
if (WRITE_FILE == TRUE)
fid = fopen('noise.pwl','w') ;
end
delT = T - T/100 ;
delTinMS = 1e3 * delT ;
delTinUS = 1e6 * delT ;
for k = 1 : N
if (WRITE_FILE == TRUE)
t(k) = 1e3 * t(k) ;
fprintf(fid, '%15.8fm\t%g\n', t(k), noise(k)) ;
fprintf(fid, '%15.8fm\t%g\n', t(k)+ delTinMS, noise(k)) ;
end
end % end for
if (WRITE_FILE == TRUE)
status = fclose(fid) ;
end
end
```

Anti-Aliasing Filter function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Anti-Aliasing Filter (2nd order Analog low-pass filter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out] = AAF(n, w, input, t)
%Resolution = 1 ; % Frequency resolution in Hertz
%Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
%N = Fs / Resolution ; % Length of input time sequence (2 to some power)
%N = 2 ^ (round(log2(N))) ;
%n = 2; % 2st order LPF for a limited GBW of the op-amplifier
%f3 = 12e3; % cut-off frequency for a limited GBW (fc0 = 2*Fch = 32 KHz)
%w = 2*pi*f3; % the angular frequency
%T = 1 / Fs ;
%t = T * linspace(0, N-1, N) ;
[z, p, k]=butter(n,w, 's'); % Zero-Pole-Gain design
lp = zpk(z,p,k); % creates a continuous-time zero-pole-gain model
tflp = tf(lp); % convert to transfer function model
% plot of the transfer function
%h = bodeplot(tflp); setoptions(h,'FreqUnits','Hz');
%setoptions(h1,'FreqUnits','Hz','PhaseVisible','off');
out = lsim(tflp,input,t); % apply the filter to an input signal
```

Band-Pass Filter function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 8th order DSP Band-Pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out] = BandPass(n, w, input, t)
%Resolution = 1 ; % Frequency resolution in Hertz
%Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
%N = Fs / Resolution ; % Length of input time sequence (2 to some power)
%N = 2 ^ (round(log2(N))) ;
%n = 4; % 8th Order BPF filter
%w = [2*pi*75 2*pi*105]; % passband range: 75Hz - 105Hz
%T = 1 / Fs ;
%t = T * linspace(0, N-1, N) ;
[z, p, k]=butter(n,w,'s'); % zero-pole-gain design
bp = zpk(z,p,k); % creates a continuous-time zero-pole-gain model
tfbp = tf(bp), % convert to transfer function model
% plot of the transfer function
% h = bodeplot(tfbp); setoptions(h,'FreqUnits','Hz');
out = lsim(tfbp,input,t); % apply the filter to an input signal
```

FFT Magnitude Computation function

```
function[out] = FFTmagnitude(Waveform)
%
% Using Hodie window to minimize spectral blurring
%
L = length(Waveform) ;
coefficients = HodieWindow(L) ;
WindowedWaveform = Waveform .* coefficients' ;
%
% Return magnitude part of FFT
%
out = abs(fft(WindowedWaveform)) ;
end
```

<u>**Hodie Window function**</u>

```
function [out] = HodieWindow(N)
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 16*131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence
N = 2 ^ (round(log2(N))) ; % 2 to some power
%
% Script to compute a N-term floating point Hodie window
% Copyright 2001 Eric Swanson
% N is the number of filter coefficients
%
window = zeros(N,1);
bottom = zeros(N,1);
top = 2.5 * ones(N,1);
index = [0:N-1]';
v = 2 * pi / N;
%
% Hodie window cosine coefficients
%
a0=0.61640321314050;
a1=0.98537119272586;
a2=0.49603771622007;
a3=0.14992232793243;
a4=0.02458719103474;
a5=0.00176604651487;
a6=0.00003158118857;
%
% Coefficients sum to N
%
for m=1:N;
n1=m-.5;
window(m,1)=a0-a1*cos(v*n1)+a2*cos(v*2*n1)-a3*cos(v*3*n1)+a4*cos(v*4*n1)-
a5*cos(v*5*n1)+a6*cos(v*6*n1);
end;
out = window;
%
%plots
%
plot(index,window,'-g',index,bottom,'-b',index,top,'-b');
%axis([0 1000 0 2.5]);
%axis off;
```

SNR Computation function

```
function [SNR] = ComputeSNR(FFT, Fs, Fo, BW)
%
% Generates a sinusoidal time sequency
%
% N is the number of samples to in the time sequence
% A1 is the peak amplitude of the sine wave
% Fs is the sampling frequency
% Fo is the frequency of the sine wave in Hertz
%
% Fhat is the normalized frequency i.e. F0/Fs
%
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence (2 to some power)
N = 2 ^ (round(log2(N))) ;
A1 = 1e-6; % Input sine wave amplitude
Fo = 45; % Frequency of input sine wave
Fhat = Fo / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SineWave = A1 * sin(K * index);
Ts = 1 / Fs ;
time = Ts * index ;

%
% Script to compute signal-noise-ratio
%
%
% Compute signal bin
%
FFT = SineWave;

L = length(FFT) ;
BW = 30;
BinSize = Fs / L ;
SignalBin= round(Fo/BinSize) ;
%
% Hodie Window confines spectrum to Signal Bin +/- 8 bins
%
LowerSignalBin = SignalBin - 8 ;
UpperSignalBin = SignalBin + 8 ;
%
% Compute end of BW bin
%
BWbin= round(BW/BinSize) ;
SumSignalSquared = 0.0 ;
SumNoiseSquared = 0.0 ;
%
% Calculate noise power preceding signal bins
%
for k = 2 : 1: (LowerSignalBin - 1)
SumNoiseSquared = SumNoiseSquared + FFT(k) * FFT(k);
end
%
```

```
% Estimate the noise power in the signal bins
%
EstNoisePwr = FFT(LowerSignalBin - 1) * FFT(LowerSignalBin - 1) ;
EstNoisePwr = EstNoisePwr + FFT(UpperSignalBin + 1) * FFT(UpperSignalBin + 1) ;
EstNoisePwr = EstNoisePwr / 2.0 ;
%
% Calculate signal power
%
for k = LowerSignalBin: 1 : UpperSignalBin
SumSignalSquared = SumSignalSquared + FFT(k) * FFT(k) - EstNoisePwr ;
SumNoiseSquared = SumNoiseSquared + EstNoisePwr ;
end
%
% Calculate noise power after signal bins
%
for k = UpperSignalBin + 1 : 1 : BWbin
SumNoiseSquared = SumNoiseSquared + FFT(k) * FFT(k) ;
end
SNR = 10 * log10(SumSignalSquared/SumNoiseSquared);
plot(time, SNR);
end
```

Time Domain Plot function

```
function[] = TimeDomainPlot(Wave, Vref, Fs, color, locn, total)
%
% Generates a sinusoidal time sequency
%
% N is the number of samples to in the time sequence
% A1 is the peak amplitude of the sine wave
% Fs is the sampling frequency
% Fo is the frequency of the sine wave in Hertz
%
% Fhat is the normalized frequency i.e. F0/Fs
%
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence (2 to some power)
N = 2 ^ (round(log2(N))) ;
A1 = 1e-6; % Input sine wave amplitude
Fo = 45; % Frequency of input sine wave
Fhat = Fo / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SineWave = A1 * sin(K * index);
Ts = 1 / Fs ;
time = Ts * index ;
%
% This script will plot waveforms associated with delta sigma ADC
%
Wave = SineWave;
L = length(Wave) ;
index = linspace(0, L-1, L) ;
Ts = 1 / Fs ;
time = Ts * index ;
%subplot(total, 1, locn) ;
plot(time, Wave) ;
end
```

Frequency Domain Plot function

```
function[] = FreqDomainPlot(FFTwaveform, Vref, Fs, color, locn, total)
%
% Generates a sinusoidal time sequency
%
% N is the number of samples to in the time sequence
% A1 is the peak amplitude of the sine wave
% Fs is the sampling frequency
% Fo is the frequency of the sine wave in Hertz
%
% Fhat is the normalized frequency i.e. F0/Fs
%
Resolution = 1 ; % Frequency resolution in Hertz
Fs = 131072 ; % Modulator sampling frequency Fs = 2^17Hz
N = Fs / Resolution ; % Length of input time sequence (2 to some power)
N = 2 ^ (round(log2(N))) ;
A1 = 1e-6; % Input sine wave amplitude
Fo = 45; % Frequency of input sine wave
Fhat = Fo / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SineWave = A1 * sin(K * index);
Ts = 1 / Fs ;
time = Ts * index ;
%
% This script will plot FFTs associated with chopper amplifier
%
FFTwaveform = SineWave;
Vref=3.6;
L = length(FFTwaveform) ;
DBFS = 20*log10((2*FFTwaveform)/(L * Vref)) ;
index = linspace(0, L/2-1, L/2) ;
deltaF = Fs / L ;
frequency = deltaF * index ;
%subplot(total, 1, locn) ;
plot(frequency, DBFS(1 : L/2)) ;
end
```