Project Design Laboratory

FALL DETECTION SENSOR

Project documentation

ANZEN project team:

Ankita Rambhal

Pallavi Praful

Md. Rakin Sarder

Faith Onuoha

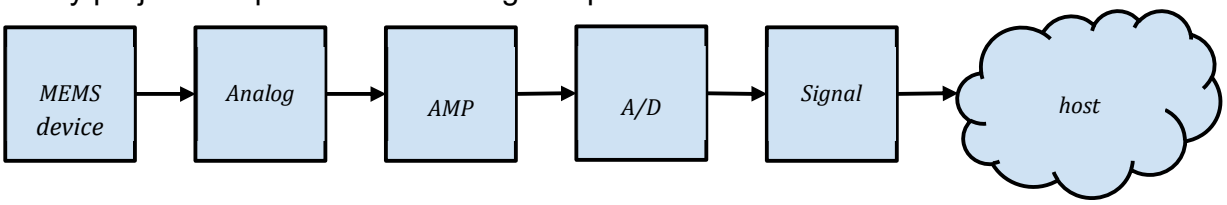
Date: 1.11.2020

1 Project objective

The overall goal of this project is the design of a complete smart system consisting of a MEMS sensor, an analog signal conditioning circuit and digital signal processing. A user specification is given for each project team. The specification parameters differ from project to project according to the specified application.

2 Design structure overview

Every project comprises the following components:



2.1 MEMS device

- a sensor type device, a cantilever with piezoresistive sensing elements
- the sensors are subjected to an acceleration loading
- **Design input:** The basic geometry structure of the MEMS devices
- **Design parameters:** Geometry sizing, the amplitude, time and direction of the gravity and/or pressure loading
- **Design task:** Size the geometry of the device, validate with simulation, create a reduced-order model of the device

2.2 Analog interface

- Signal conditioning circuit
    - Can be an operational amplifier which converts the change in resistance of the piezoresistor to a voltage which loads a voltage controlled oscillator
- **Input:** idealized circuit models, circuit schematics
- **Design parameters:** gain, bandwidth, offset, noise, current consumption, output resistance.
- **Design tasks:**
    - Determine a set of design parameters which fulfil the customer specified operation.
    - Sizing of the circuits and validation by simulations

2.3 Signal processing

- The signal processing circuit monitors the frequency of the VCO. Based on the acquired data it makes a decision whether an interrupt should be sent to the host microprocessor or not.
- **Input:** All VHDL codes presented in the video tutorials may be reused.
- **Design parameters:** Initial frequency, input frequency range
- **Design task:**
    - Determine a set of design parameters which fulfil the customer specified operation.
    - Design the signal processing circuit in VHDL, validate with simulation and synthesize the circuit

3 Customer specification

For elderly individuals and those vulnerable to falls, falls are a serious concern. Of the population aged 65 years and over, one-third to one-half have suffered falls [1]. Unfortunately, falls are a dynamic phenomenon, and they are the result of any ongoing illness and predict potential impairment. They are triggered by environmental and dynamic equilibrium interactions, which are determined by the consistency of sensory feedback, internal processing, and motor responses [2]. According to a report published by the WHO in 2018 [3],

- Falls are the second leading cause of accidental or unintentional injury deaths worldwide.
- Around 646 thousand people lose their lives from falls around the world, of which over 80% are developing and under-developed nations.
- The greatest number of fatal falls are suffered by adults older than 65 years of age.
- 37.3 million falls that are severe enough to require medical attention occur each year.
- Prevention strategies should emphasize education, training, creating safer environments, prioritizing fall-related research and establishing effective policies to reduce risk.

Even a drop that does not lead to injury may have significant implications. A downward spiral of self-imposed reduced activity can be generated by psychological stress and fear of falling, leading to a loss of strength, endurance, and agility, thus raising the likelihood of potential falls and injuries [4].

That is why fall detection is extremely significant in the nursing and care of the old people. The primary challenges to tackle are:

- Monitoring and learning the **P**osture, **M**otion and **O**rientation (PMO) of the elderly/patient
- Detecting any sudden anomaly in the PMO and act instantly

- Creating auditory alert system to let people nearby regarding the subject's fall

Our proposed system "Anzen" is a smart fall detector and prevention belt which will detect a person falling and immediately trigger a local alarm system to alert nearby people regarding a subject's fall.

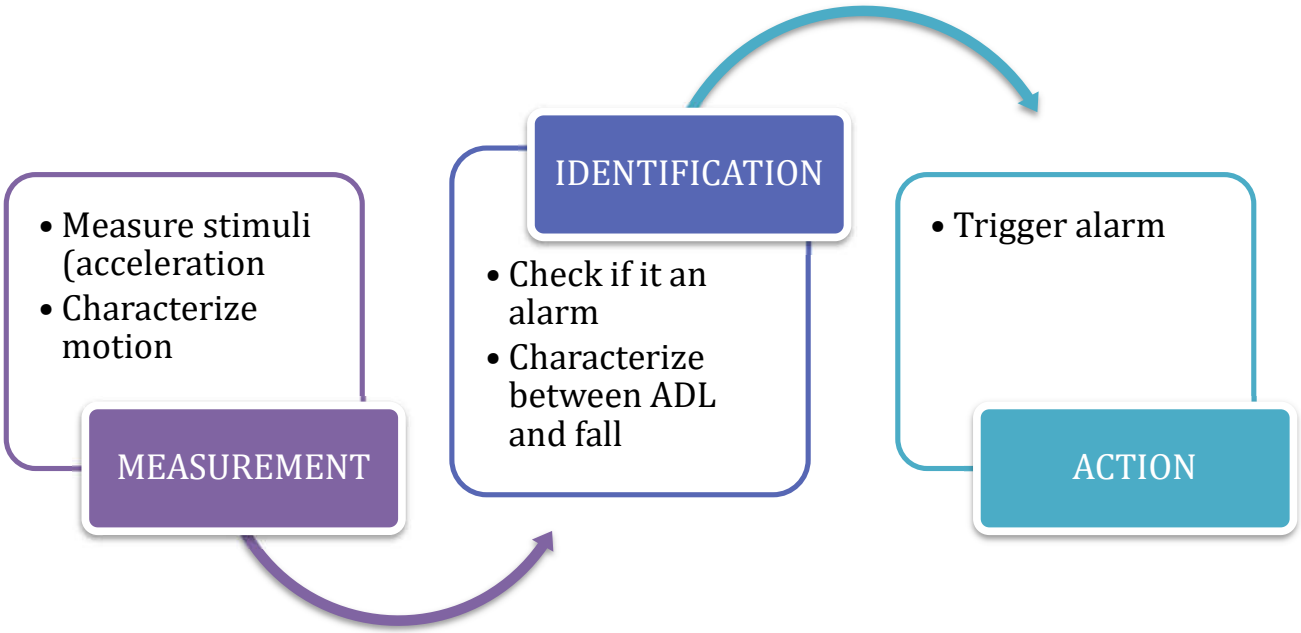


Figure 1: Anzen fall detection system overview

The salient features of our proposed system are:

- Setup a threshold acceleration voltage which will differentiate between a possible fall case and Activities of Daily Living (ADL).
- Detects any falling scenario for 170ms~220ms.
- Trigger an alarm once a fall event is identified and alert others in the vicinity.

One of the biggest considerations before starting the design of the fall detection system is the choice of placement of the sensor. This is because the sensor response depends upon the interpersonal differences of the user. The interpersonal difference is directly proportional to:

- Gender of the user
- Unique postures made by a user
- User physical characteristics (such as obesity, thinness etc.)

While the interpersonal differences increase due to the subject's attributes, the performance accuracy of the fall detection decreases. Therefore, to limit the interpersonal differences at a

minimum level, the sensor position is chosen. A study based on [5] shows the accuracy based on sensor placement of a single sensor-based solution (Figure 2). Based on the study, we are choosing waist as the suitable sensor placement, since it provides the highest accuracy, and this region is the closest to the user's centre of gravity. Our chosen position is also backed by other notable works [6, 7]. In terms of user comfort, waist position is also suitable, as it has been backed by different works [8] and existing products in the market [9, 10].

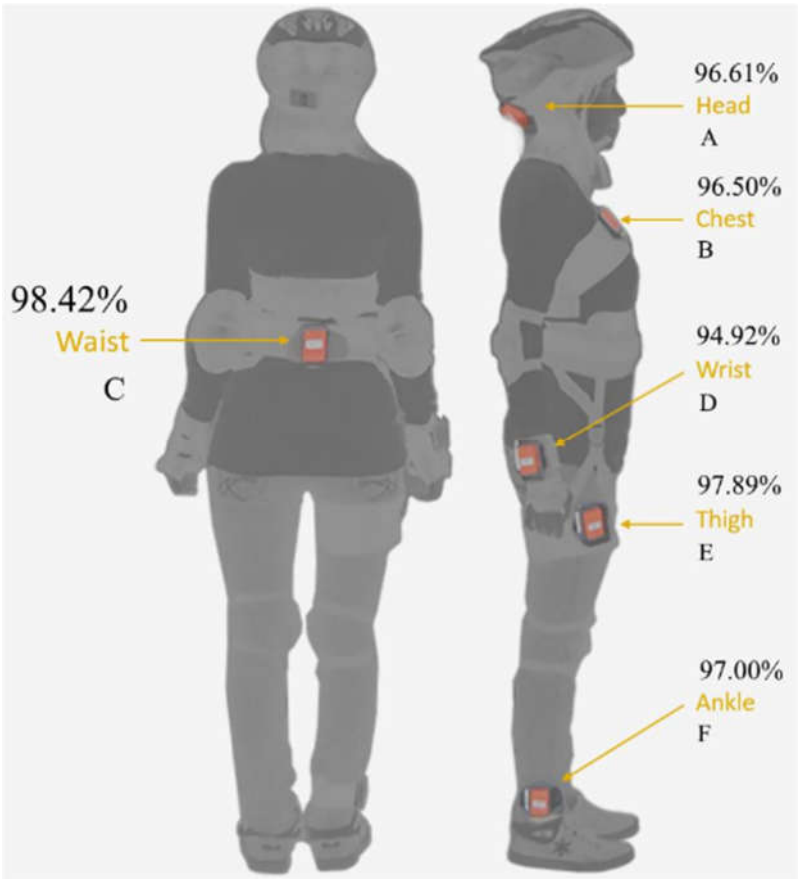


Figure 2: Accuracy of fall detection for different placements of a detection device [5]

"Anzen" belts are to be worn by the user above the hips, near the belly region. From an engineering point of view, the belt consists:

- Accelerometer
- Buzzer
- A central processing unit (which integrates the sensors, airbag driver module and the buzzer)

To achieve to our desired project goal, the whole process architecture can be summarized into three design sections:

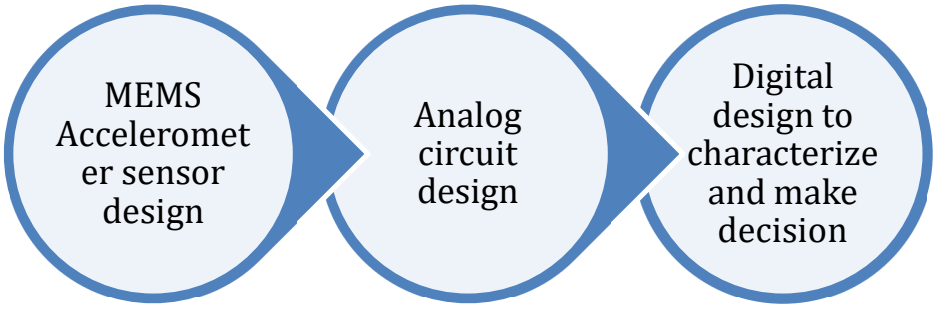


Figure 3: Design sections of Anzen

Since our project is operating based on environmental stimuli and threshold-based detection, the system is suitable for real-time implementation[11]. However, to limit the design complexity, we are limiting our system based on only one MEMS acceleration sensor. While this provide some clarity and simplicity to our design process, we are limited in characterizing some ADLs and fall events which are significantly dependent on the orientation of the device [12].

4 Design of the project workflow

List of Work Packages (WPs)

WP no.	WP title	WP leader	SUM workload
1	Behavioral modeling in system level	Pallavi Praful	52
2	Analog circuit design	Ankita Rambhal	40
3	Digital circuit design	Md. Rakin Sarder	40
4	Presentation of the work	Faith Onuoha	12

Timing in workdays:

WP No.	Starts	Duration	Ends
Kickoff meeting	D0		
1	D1	33	D34
Progress meeting	D35		
2	D35	18	D53
3	D35	18	D53
4	D54	6	D60
Project evaluation	D61		

Deadlines:

Kickoff meeting	09 October 2019 12:15
WP1 submission	12 November 2019 00:00
Progress meeting	13 November 2019 14:15
WP2 submission	1 December 2019 00:00
WP3 submission	1 December 2019 00:00
Progress meeting	2 December 2019 10:15
WP4 submission	8 December 2019 00:00
Final meeting	9 December 2019 10:15

5 Work Package 1: Behavioral modelling in system level

5.1 Overview

WP Leader	Pallavi Praful	Rakin Sarder	Ankita Rambhal	Faith Onuoha
Pallavi Praful	30	5	3	14
SUM Workload:	52	**Duration:**	33	

Objectives

- Understanding the customer specification and create the system specification
- Create a behavioral model for the MEMS device, the analog and the digital circuits

Tasks

- Carrying out hand calculations based on the customer specification to determine the basic geometry of the MEMS devices. Note the different stimuli in case of front and side impact.
- Performing FEM simulations of the proper sized MEMS devices
    - Test loads, Element loads (pressure and gravity)
    - Modal analysis
- Producing the reduced-order models (ROM) of the MEMS devices
- Identifying typical working conditions (in case of front and side impact)
- Perform master node displacement analysis under standard working conditions
    - Identify the moving margins of the device
- Transient analysis: identify the required time resolution
- Calculate the piezo resistors resistance-change based on the master node displacements
- Calculate the required parameters of the amplifier stage
- Calculate the required parameters of the VCO
- Define the required functionalities of the digital data processing unit.
- Perform an integrated system testing and validate the parameters

Deliverables

- D1.1 System specification
- D1.2 MEMS model parameters, ROM parameters
- D1.3 Proposed MEMS technology and process steps
- D1.4 System-level testing results
- D1.5 Module specifications

5.2 Work Package 1: Project implementation steps

For any engineering problem or a multi-domain problem that needs to be solved by engineering, the first and the most important task is to analyze the problem statement. In our case, our primary goal is to analyze human (especially elderly) activities that requires motor functionalities, postures, and locomotion.
On the other hand, to obtain suitable design parameters and a conceptual prototype of any devices, analysis of existing systems and studies provide significant insights regarding the (i) decision making, (ii) design choice and approach, (iii) static and dynamic response, and (iv) suitability in real-world context.
For the purpose of the project, we are analyzing SisFall dataset, a public available dataset consisting fall and movement data of different ADLs and falls[13]. The dataset has been published by SISTEMIC, Faculty of Engineering, Universidad de Antiquia UDEA under a CCA 4.0 International License. The dataset consists of 4510 files of different activities, including falls. The data was acquired via a set of sensor modules, which consists two triaxial accelerometers (ADXL345 and MMA8451Q) and one gyroscopic sensor (ITG3200). The dataset consists of the following features:

- Video streams recorded for each ADLs and falls which corresponds to the database
- Total participants between age 19-30 (young adults): 23
- Total participants between age 60-75 (elders): 15
- Each file consists a single activity
- No. Of ADLs measured: 19 (Table 1)
- No. Of falls measured: 15 (Table 1)
- No. of trials per falls: 5
- Time frame recorded for each fall: 15s
- No. Of samples per trials: 3000
- No. of trials per ADLs: 5 (for 14 activities), 1 (for 5 activities)
- Time frame recorded for each ADLs: 12s~100s

One of the main reasons of choosing this dataset for our project is because most of the available solutions in the market and datasets available for research lacks the records of real falls with elderly, and since most of them have been tested under a controlled environment with constraints, often they lack accuracy in their approaches when they are institutionally tested or implemented on elderly people. The SisFall research's primary purpose was to mitigate these issues, and to provide researchers and new product developers a comprehensive and accurate dataset to work on[14].

Although humans are not limited to any certain activity set, almost all the regular activities of a human can be characterized into a set. The set of activities which a human does in day to day life are called Activities of Daily Living (ADL). The physical functions that fall within ADL can define almost all types of activities. Thus, ADL classification is the fundamental basis for detecting falls. Because any activity which is not present in the ADL can be classified as a fall event. Table [i] shows different types of ADLs and falls based on the SisFall dataset. The

time of fall values was measured from the video footages using Corel VideoStudio X10. We can see for each fall, a time frame has been noted in the table. These timeframes have been approximately analyzed from the video streams attached to each fall. Timestamps from the starting of the fall to the timestamp when the fall event completed was noted, and the difference between these two values has been recorded. The time of fall varied from participants to participants within the found range based on the interpersonal differences.

Table 1: Classification of ADLs and falls with time of falls

	Sl. No	Events	Time of fall
Activities of Daily Living (ADLs)	1	Walking slowly	-
	2	Walking quickly	-
	3	Jogging slowly	-
	4	Jogging quickly	-
	5	Walking upstairs and downstairs slowly	-
	6	Walking upstairs and downstairs quickly	-
	7	Slowly sit in a half height chair, wait a moment, and up slowly	-
	8	Quickly sit in a half height chair, wait a moment, and up quickly	-
	9	Slowly sit in a low height chair, wait a moment, and up slowly	-
	10	Quickly sit in a low height chair, wait a moment, and up quickly	-
	11	Sitting a moment, trying to get up, and collapse into a chair	-
	12	Sitting a moment, lying slowly, wait a moment, and sit again	-
	13	Sitting a moment, lying quickly, wait a moment, and sit again	-
	14	Being on one's back change to lateral position, wait a moment, and change to one's back	-
	15	Standing, slowly bending at knees, and getting up	-
	16	Standing, slowly bending without bending knees, and getting up	-
	17	Standing, get into a car, remain seated and get out of the car	-
	18	Stumble while walking	-
	19	Gently jump without falling (trying to reach a high object)	-
Fall	1	Fall forward while walking caused by a slip	1~2 sec
	2	Fall backward while walking caused by a slip	1~2 sec
	3	Lateral fall while walking caused by a slip	1~2 sec
	4	Fall forward while walking caused by a trip	1~2 sec
	5	Fall forward while jogging caused by a trip	1~2 sec
	6	Vertical fall while walking caused by fainting	1~2 sec
	7	Fall while walking, with use of hands in a table to dampen fall, caused by fainting	1~2 sec
	8	Fall forward when trying to get up	1~3 sec
	9	Lateral fall when trying to get up	1~3 sec
	10	Fall forward when trying to sit down	1~3 sec
	11	Fall backward when trying to sit down	1~3 sec
	12	Lateral fall when trying to sit down	1~3 sec
	13	Fall forward while sitting, caused by fainting, or falling asleep	1~3 sec
	14	Fall backward while sitting, caused by fainting, or falling asleep	1~3 sec
	15	Lateral fall while sitting, caused by fainting, or falling asleep	1~3 sec

5.3 Required dynamic response of the sensor

As already mentioned in the customer specification, we are designing our fall detection system based on a single triaxial sensor. The SisFall dataset used two accelerometers, and one gyroscope. For the sensor data analysis, we are only considering the data from the ADXL345 accelerometer sensor. ADXL345 sensor has the following specification:

Table 2: ADXL345 sensor specification

Resolution	13 bits
Range	+-16g
Measuring values	Ax (-235 to +270)
	Ay (-240 to +260)
	Az (-240 to +270)

The first consideration to design our system was to analyze the speed of sensor data acquisition (acquisition sensitivity of data) by the SisFall system. The acquisition sensitivity contributes to the system accuracy, at the same time controls the decision-making stage of the system.

5.3.1 Equations analysis

The data included in the SisFall dataset contains the raw data from their sensor system. To convert the raw data of ADXL345 into acceleration in g, the equation is:

$$a(g) = \frac{2*Range}{2^{Resolution}} * RAD \quad (1)$$

where RAD is the raw acceleration data.

In the SisFall research[14], they have proposed a threshold-based system with a non-linear classification feature, which is combined with a Kalman filter and a periodicity detector in order reduce false fall detections. Since we are proposing a simpler approach with reduced complexity, we are choosing the following parameters and derived values as our thresholds:

- Axial accelerations,
- RMS (Root-mean-square) of the acceleration,
- Pitch of the motion, and
- Roll of the motion

The algorithm chosen for our fall detection is given by Figure 4.

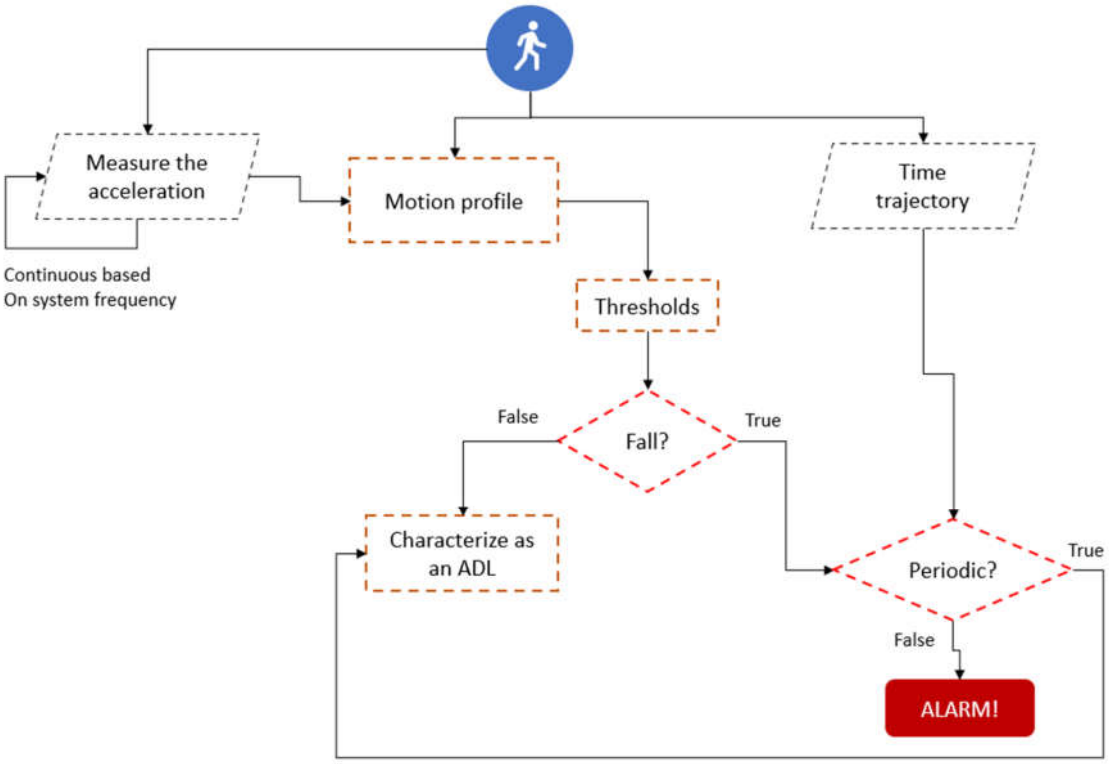


Figure 4: Fall detection algorithm chosen for the system in details

The choice of the thresholds has been backed by the following works [12, 15, 16]. The RMS of the accelerations can be found from the following equation:

$$\alpha = \sqrt{{A_x}^2 + {A_y}^2 + {A_z}^2} \tag{2}$$

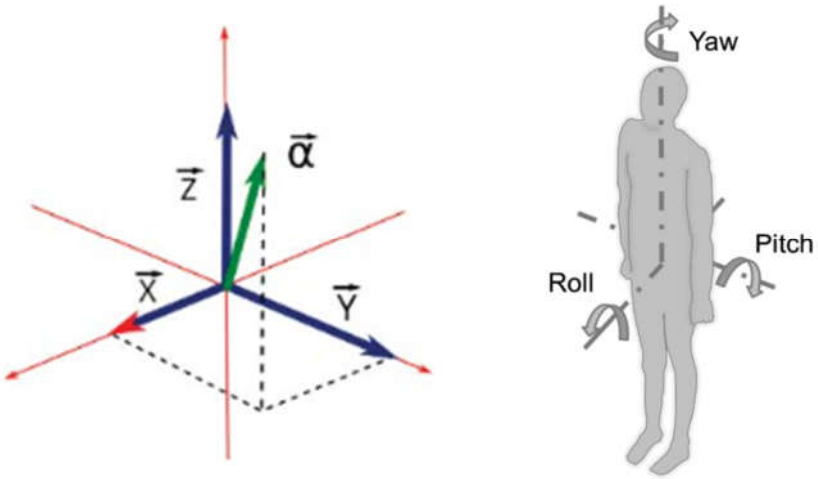


Figure 5: The RMS and the orientation angles of human body [12]

The pitch (θ) can be calculated by:

$$\theta = \tan^{-1}\left(\frac{A_x}{\sqrt{{A_y}^2+{A_z}^2}}\right) \quad (3)$$

The roll value (Φ) can be calculated by:

$$\Phi = \tan^{-1}\left(\frac{A_y}{\sqrt{{A_x}^2+{A_z}^2}}\right) \quad (4)$$

5.3.1.1

5.3.1.2 Derivation of thresholds from video and sensor data analysis:

To derive the optimum threshold values of acceleration, time frame, RMS, pitch and roll we considered these 5 fall events:

- Fall forward while walking caused by a slip
- Fall backward while walking caused by a slip
- Fall forward when trying to sit down
- Fall backward when trying to sit down
- Fall backward while sitting, caused by fainting, or falling asleep

To analyze the time frame of fall, we are analyzing the video footages for the falls from the SisFall project (Table 3). The project consists only one set (of one subject) of publicly available footages which demonstrates different ADLs and fall events. We used Corel VideoStudio X10 to capture timestamp up to millisecond level of the starting and ending of the fall event.

Figure 6: An example method of capturing the time frame of fall using the software

Table 3: Timestamp values and the time frame of fall captured from the video footages of the falls using the software

Fall type	Starting time (in sec)	Ending time (sec)	Time frame of fall (sec)
Fall forward while walking caused by a slip	14.19	16.00	1.81
Fall backward while walking caused by a slip	13.13	14.09	0.96
Fall forward when trying to sit down	10.02	11.07	1.05
Fall backward when trying to sit down	9.10	11.01	1.91
Fall backward while sitting, caused by fainting, or falling asleep	8.29	11.23	2.94

We measured the remaining thresholds considering the acceleration data within these time frames added with some additional time before and after the fall (to consider the previous/next motion). We extracted the data points from the corresponding data set and calculated the results. Table 4 shows the required values found and derived during these time frames. The equations mentioned above was used for the calculation. We have additionally measured the standard deviation of Az and RMS to determine the lower bound maximation for the acceleration threshold. Based on the method, the lower bound of our acceleration threshold along Z-axis is 0.9 g, and for RMS it is 1.5853g.

Table 4: Threshold values measured and calculated during the time frame of falls

Fall type	Mean_ Ax	Max_ Ax	Mean_ Ay	Max_ Ay	Mean_ Az	Max_ Az	STD_Az	RMS_ mean	RMS_ max	RMS_S TD	Pitch_ Mean	Pitch_ Max	Roll_ Mean	Roll_ Max
Fall forward while walking caused by a slip	0.34	7.36	0.635	3.617	0.435	1.980	0.343	1.048	8.015	0.61	-10.66	45.457	-43.12	83.21
Fall backward while walking caused by a slip	0.206	3.109	0.433	8.55	0.744	5.058	0.461	1.035	10.410	0.738	-7.959	54.168	-22.472	79.671
Fall forward when trying to sit down	0.342	5.47	0.598	5.523	0.611	9.929	0.660	1.077	11.149	0.634	-18.652	17.396	-31.690	88.827
Fall backward when trying to sit down	0.126	1.890	0.557	4.941	0.596	4.765	0.437	1.002	6.803	0.385	3.976	52.479	-37.377	50.238

Fall backward while sitting, caused by fainting, or falling asleep	0.120	1.890	0.639	4.941	0.526	4.765	0.487	1.007	6.803	0.393	2.934	52.479	-44.387	50.238

5.3.1.3 Summary

The result for finding the approximate thresholds derived from the data set and the video are given below in Table 5.

Table 5: Summary of the chosen threshold values for the project

Parameter	Threshold
Axial acceleration	0.9g
RMS acceleration	1.5853g
Time frame of fall	1.734 sec
Pitch range for fall	$\theta > 45^o$ or $\theta < -45^o$
Roll range for falls	$-45^o < \Phi < 45^o$

5.3.1.4 Case study: Elder

To verify the derived thresholds for fall detection with real-life cases, we are using two SUSs (Subject-Under-Study), from the SisFall research. The description of our first test subject is as follows:

Table 6 Details of the first test subject

Gender	Male
Age	60 years old
Height	173 cm
Weight	79 kg

ADL comparison: Comparisons of the chosen threshold against some primary ADLs of the subject are given from Fig 7. to Fig 10. From the figures, we can see that the apart from some minor anomalies, in most cases the threshold values could satisfy the activity conditions.

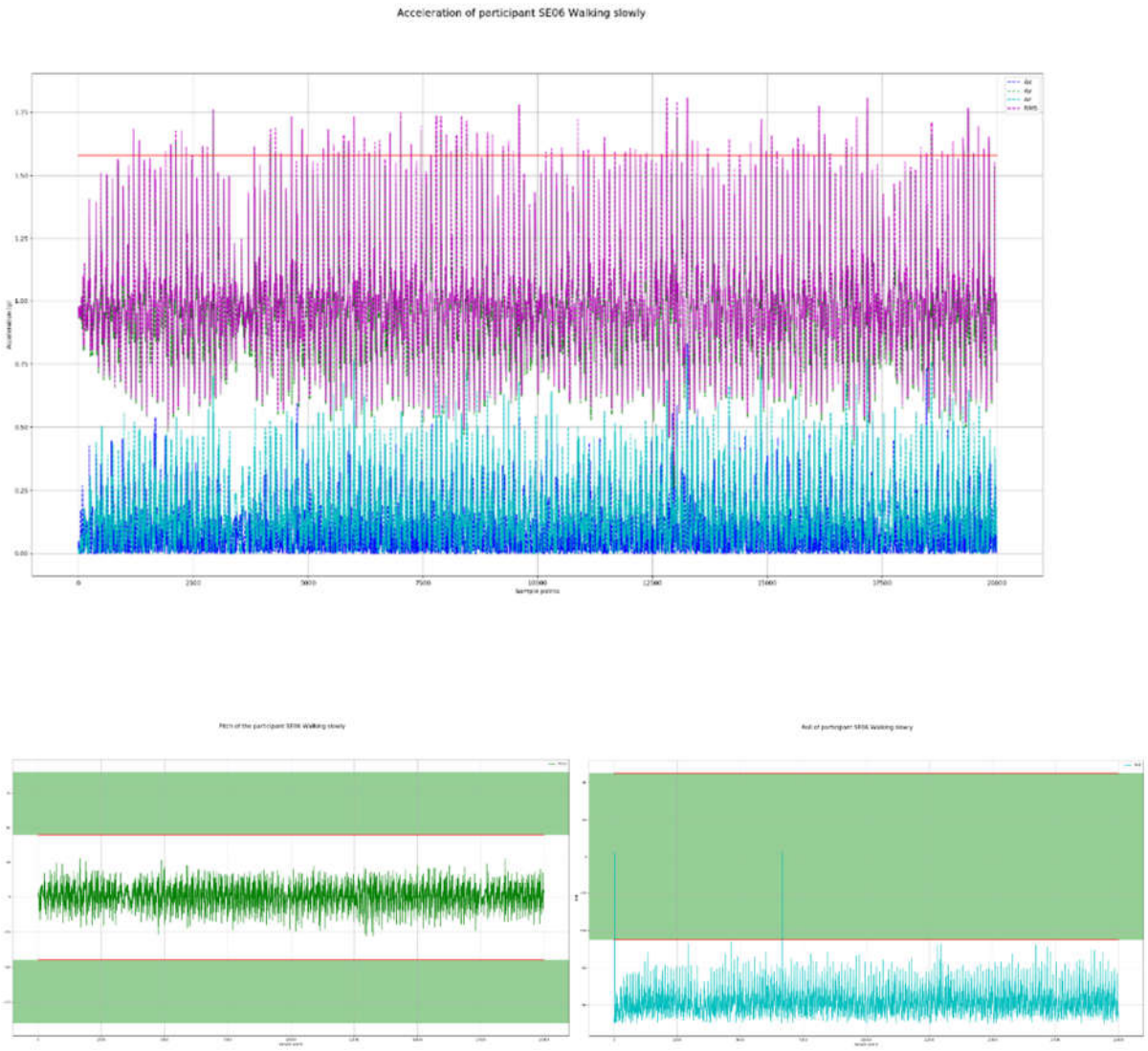


Figure 7(a-c) Acceleration, RMS, Pitch and Roll value of the elderly subject walking slowly. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

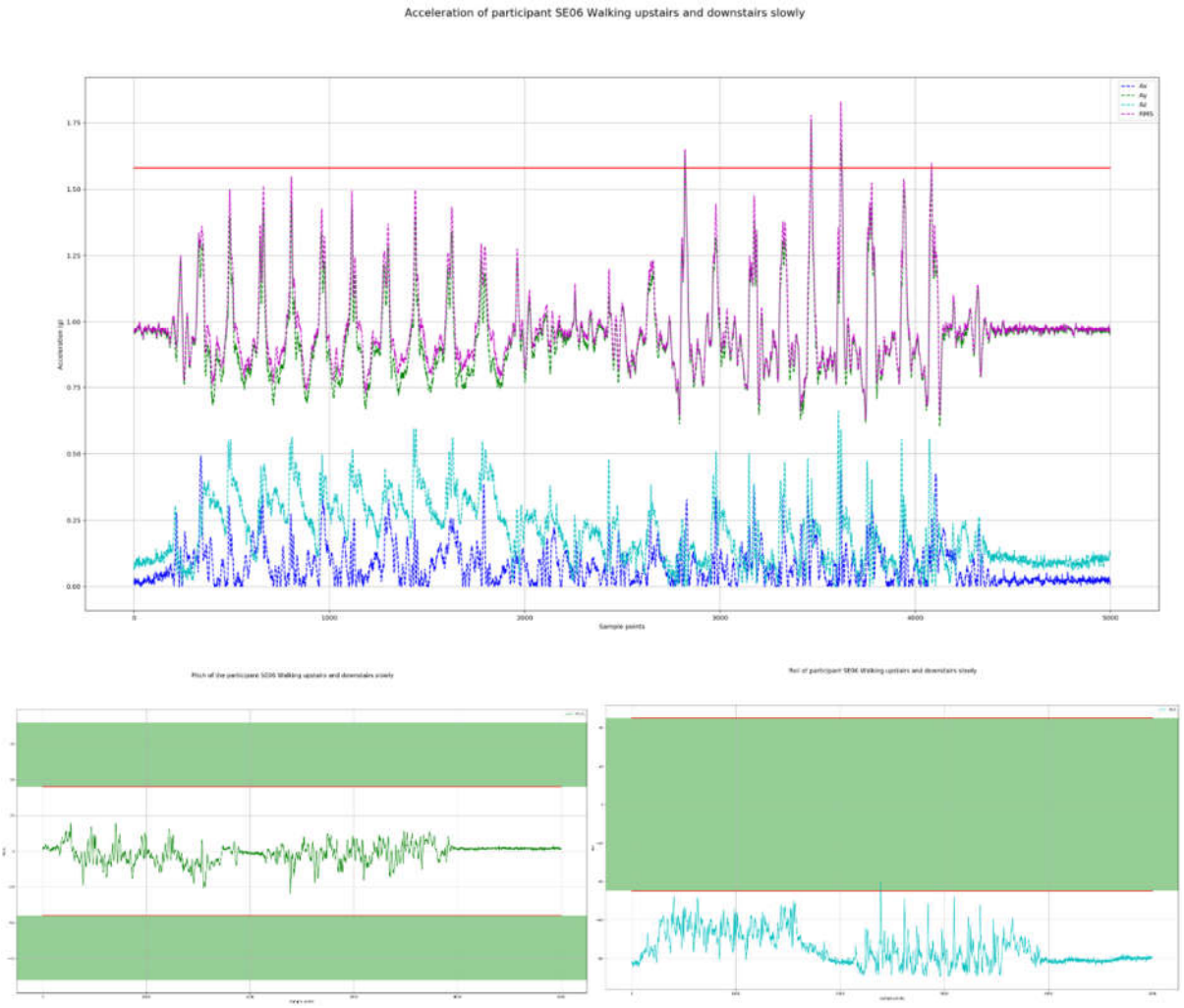


Figure 8 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly when walking upstairs and downstairs slowly. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

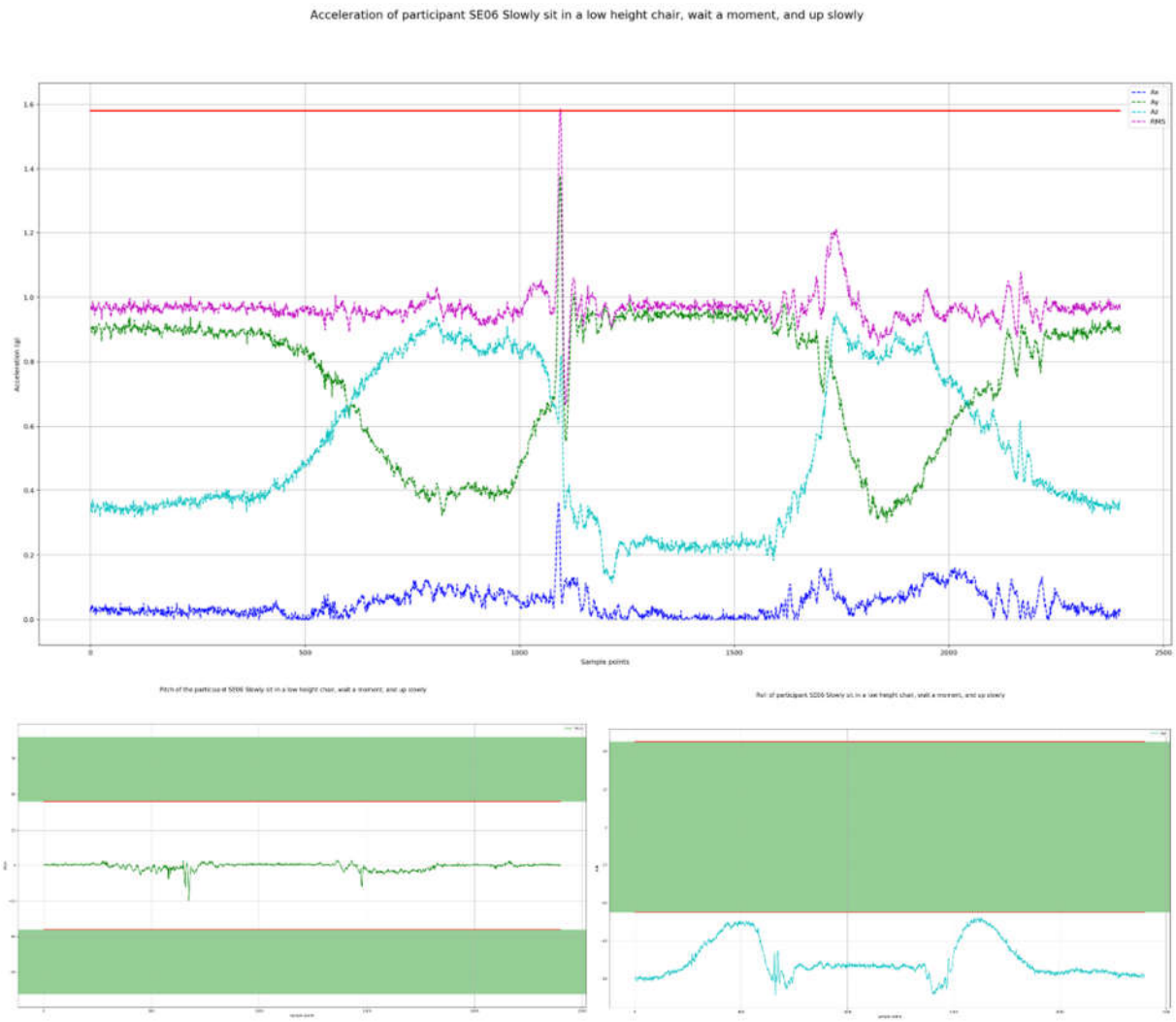


Figure 9 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly subject slowly sit in a low height chair, wait a moment. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

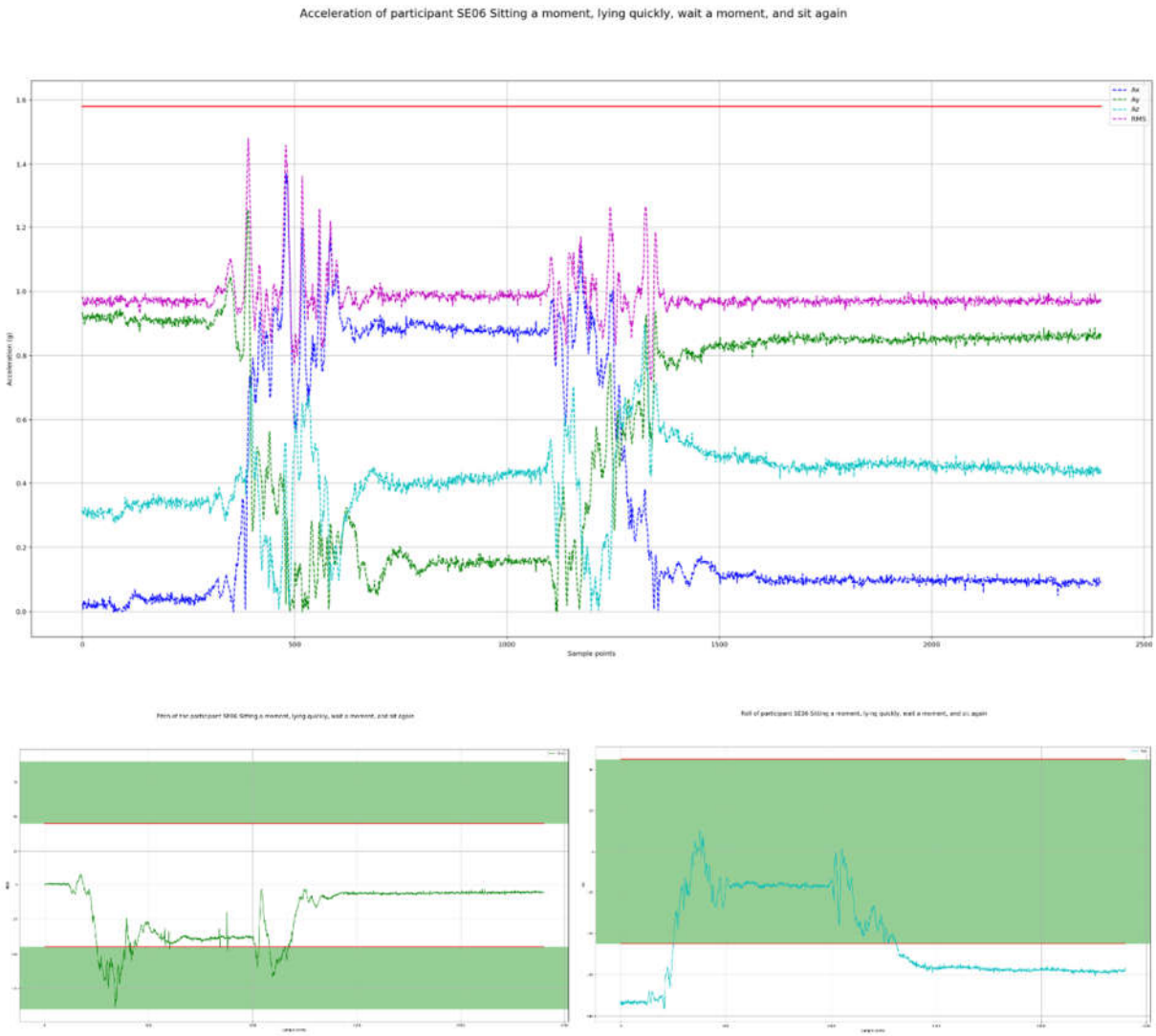


Figure 10 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly subject sitting a moment, lying quickly, wait & sit again. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

In this ADL case, we can see that the angular values went beyond the threshold, during the time the subject was lying. But the acceleration threshold level was not surpassed, therefore the system will not characterize this as a fall event. However, this is one of the other cases that might give some false-positive inputs.

Fall comparison: Comparisons of the chosen threshold against some primary falls of the subject are given from Fig 11 to Fig 15. From the figures, we can see that the apart from some minor anomalies, in most cases the threshold values could satisfy the activity conditions.

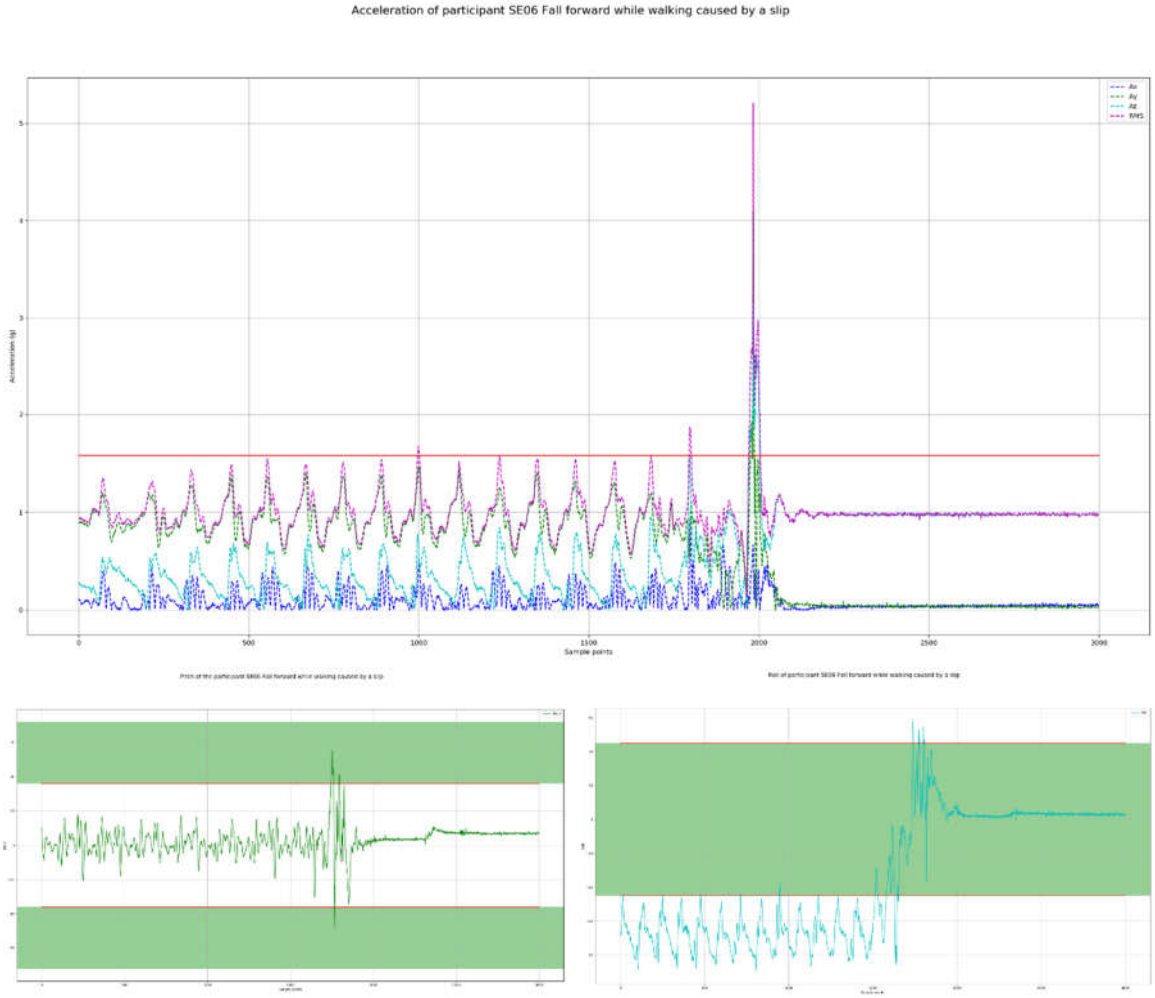


Figure 11 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly Fall forward while walking caused by a slip. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

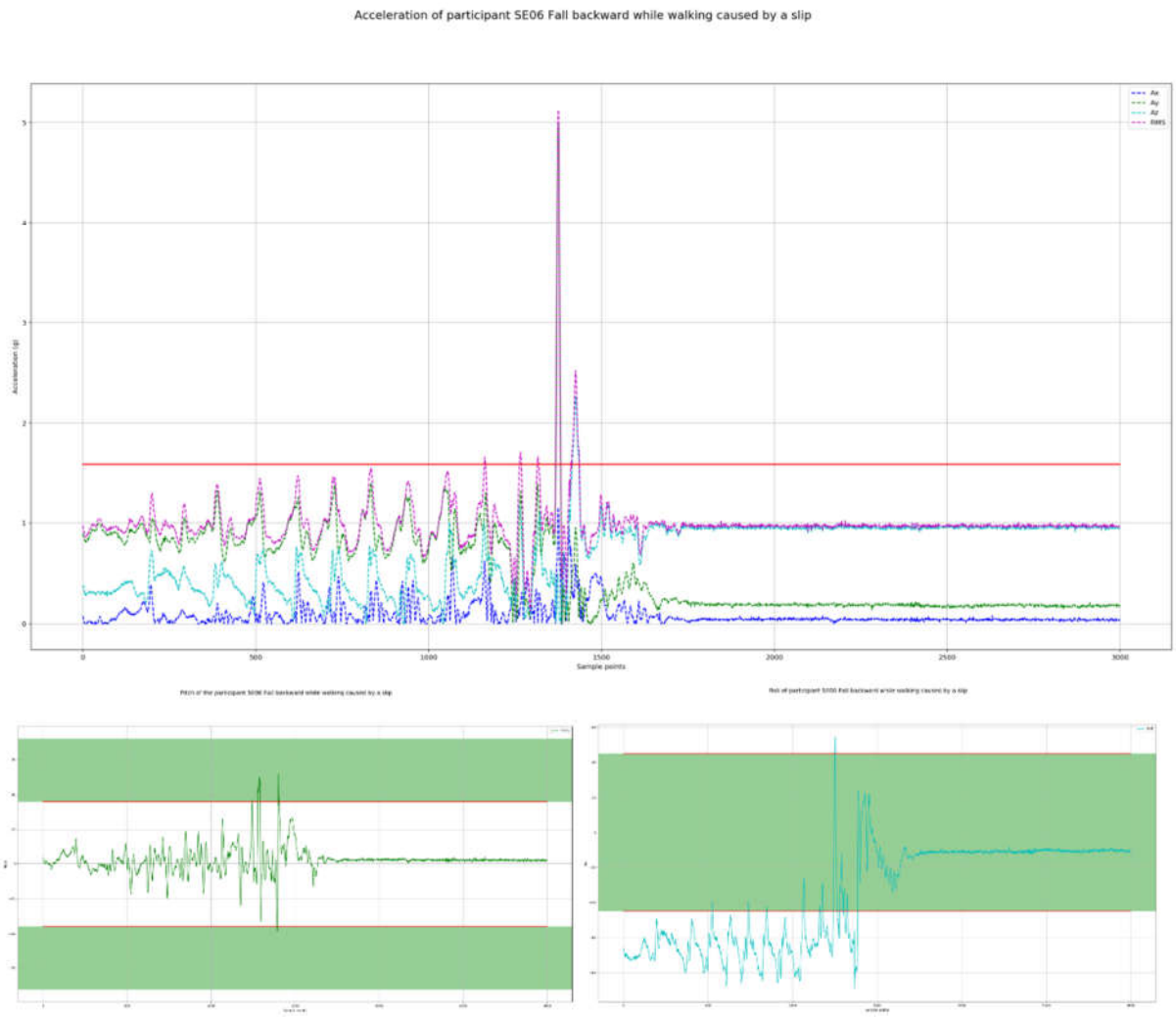


Figure 12 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly Fall back while walking caused by a slip. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

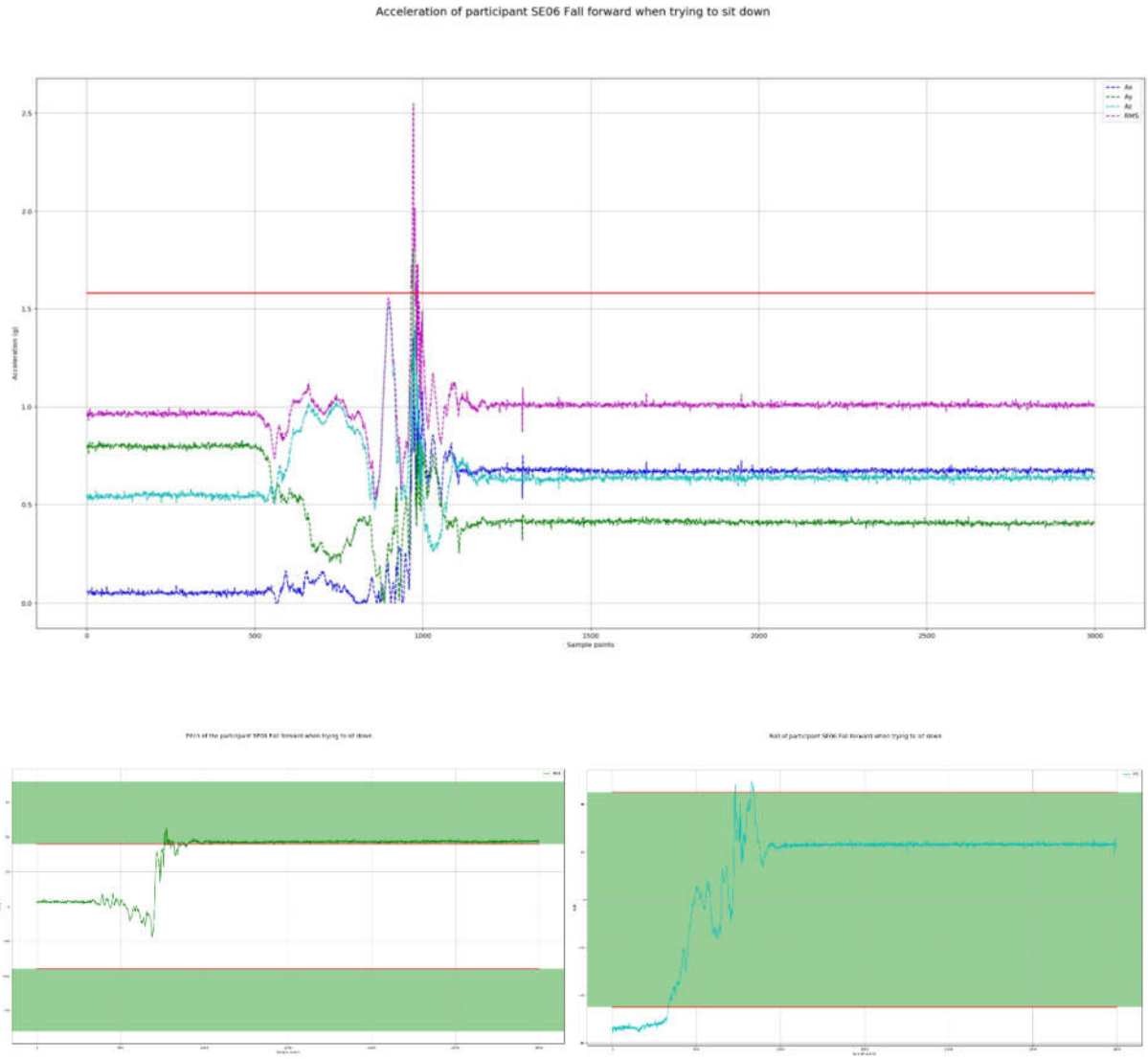


Figure 13 (a-c) Acceleration, RMS, Pitch and Roll value of the elderly fall forward when trying to sit down. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

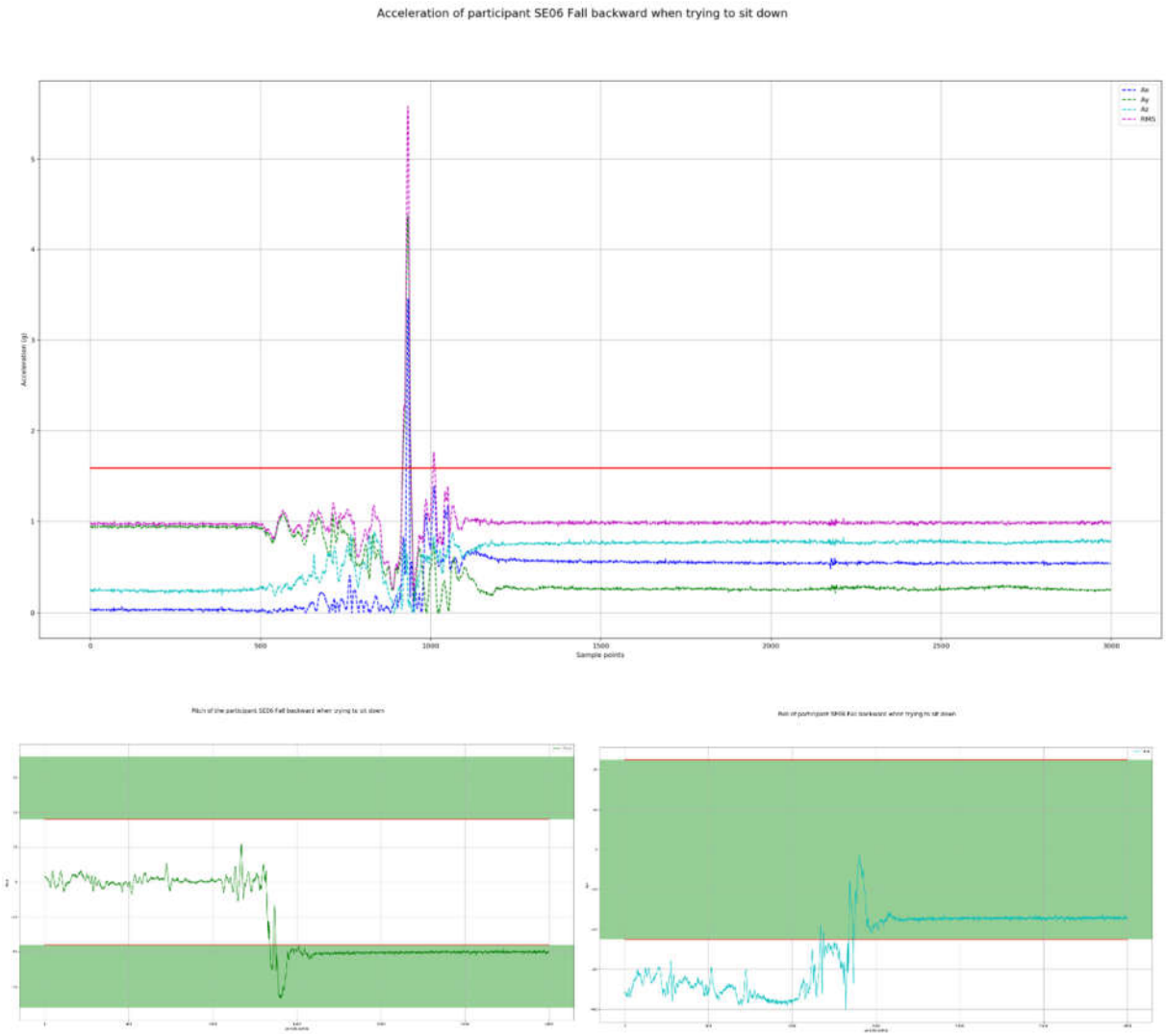


Figure 14(a-c) Acceleration, RMS, Pitch and Roll value of the elderly fall backward when trying to sit down. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

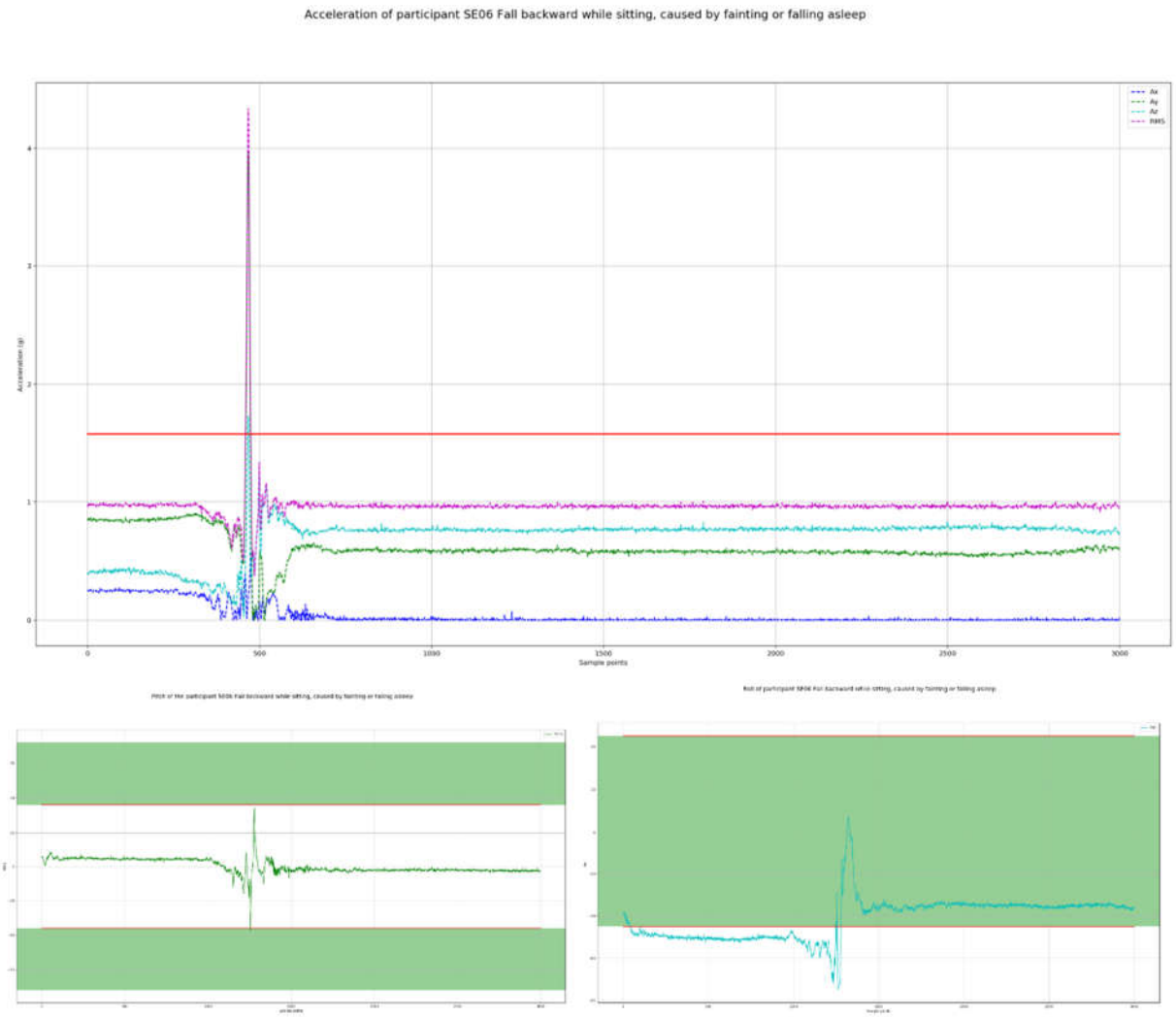


Figure 15 Acceleration, RMS, Pitch and Roll value of the elderly fall backward while sitting, caused by fainting or falling asleep. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

5.3.1.5 Case study: Young Adult

For our second test to verify our fall-detecting thresholds, we are choosing a young adult. The description of the second test subject is as follows:

Table 7 Details of the second test subject

Gender	Female
Age	30 years old
Height	150 cm
Weight	42 kg

Comparisons of the chosen threshold against some primary falls and primary ADLs of the subject are given from Fig 16 to Fig 19. From the figures, we can see that the apart from some minor anomalies, in most cases the threshold values could satisfy the activity conditions (next page).

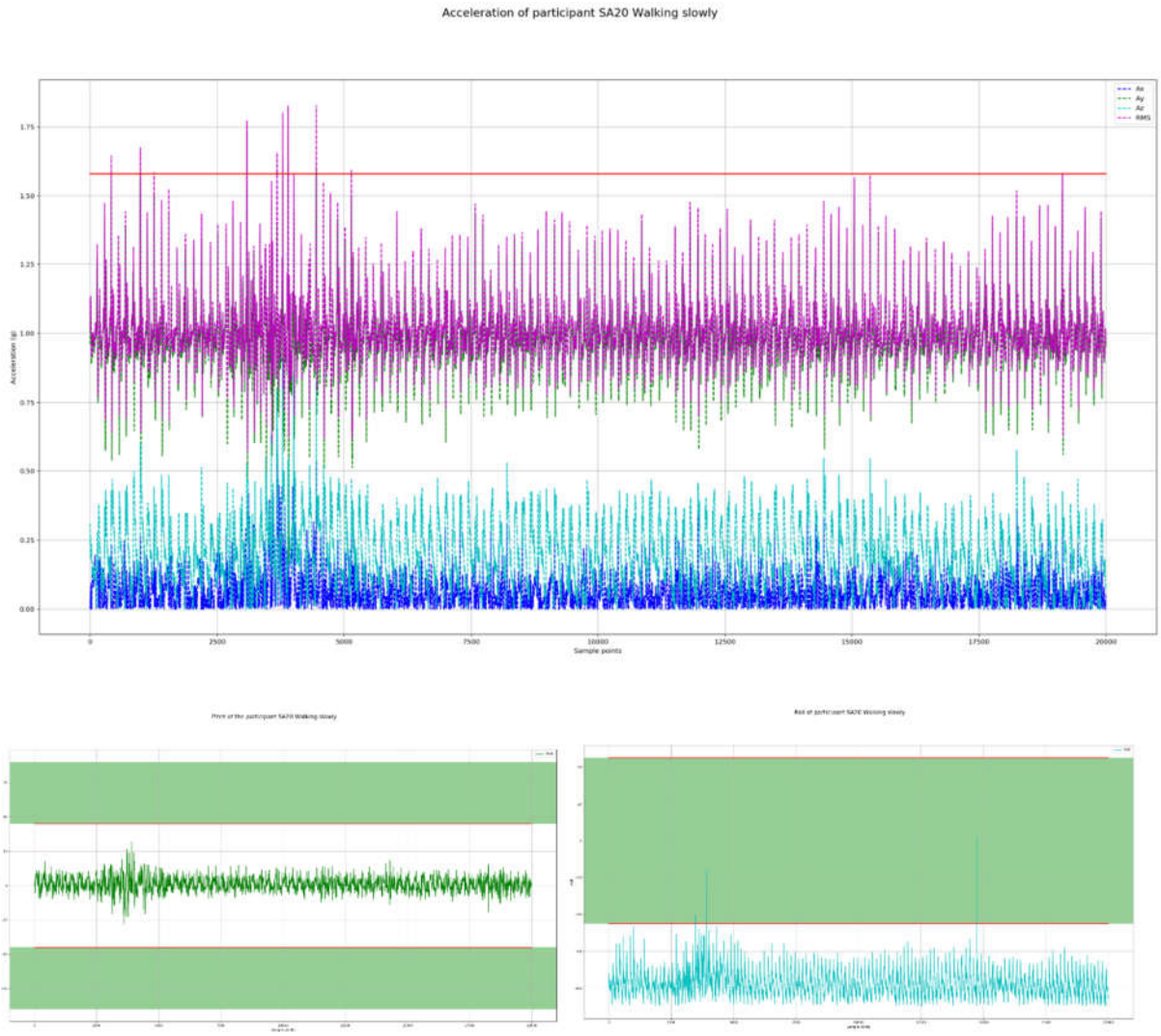


Figure 16 (a-c) Acceleration, RMS, Pitch and Roll value of the adult subject walking slowly. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

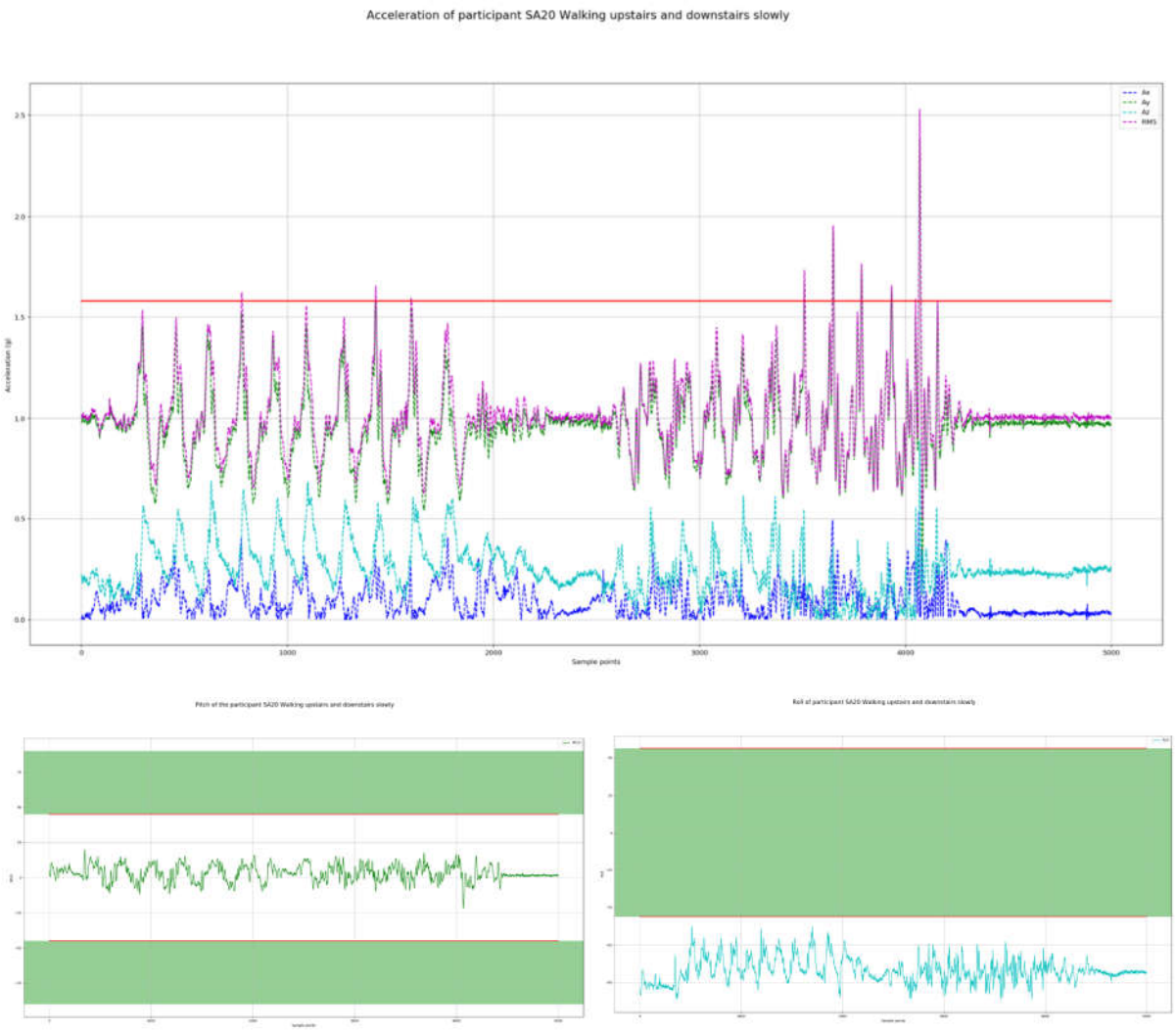


Figure 17 Acceleration, RMS, Pitch and Roll value of the adult when walking upstairs and downstairs slowly. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

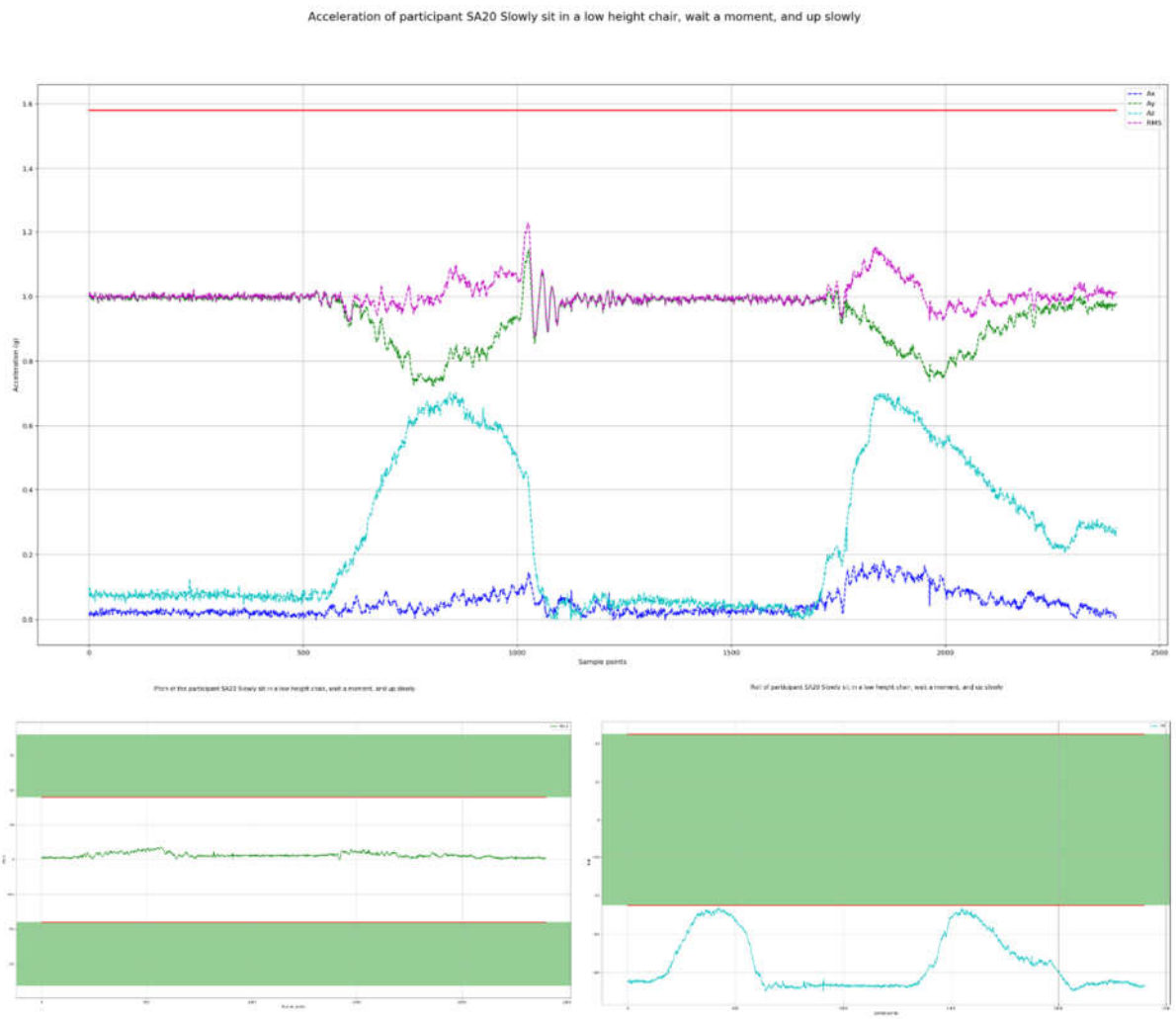


Figure 18 Acceleration, RMS, Pitch and Roll value of the adult subject slowly sit in a low height chair, wait a moment. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

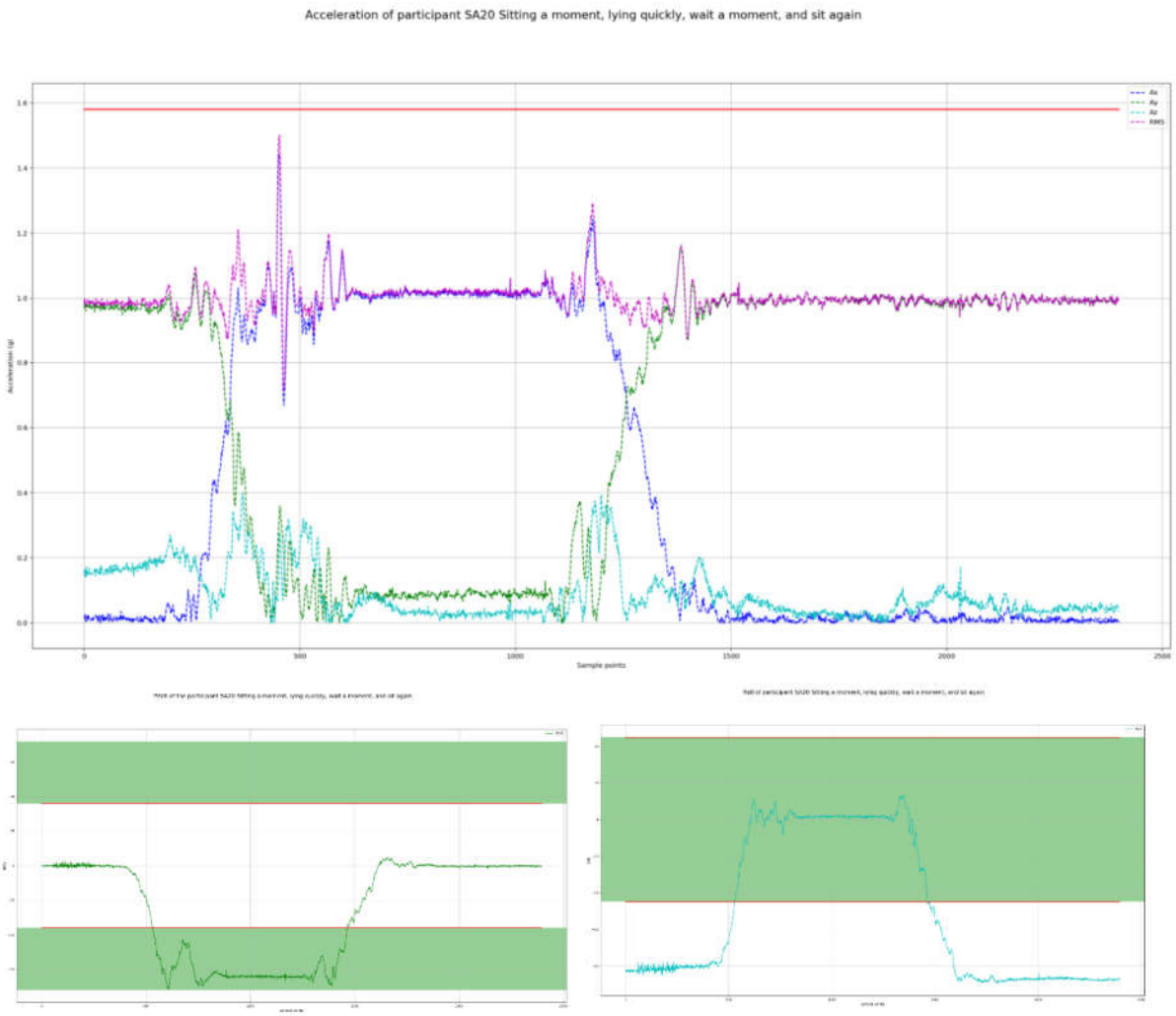


Figure 19 (a-c) Acceleration, RMS, Pitch and Roll value of the adult subject sitting a moment, lying quickly, wait & sit again. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

Fall comparison: Comparisons of the chosen threshold against some primary falls of the subject are given from Fig 20 to Fig 24. From the figures, we can see that the apart from some minor anomalies, in most cases the threshold values could satisfy the activity conditions.

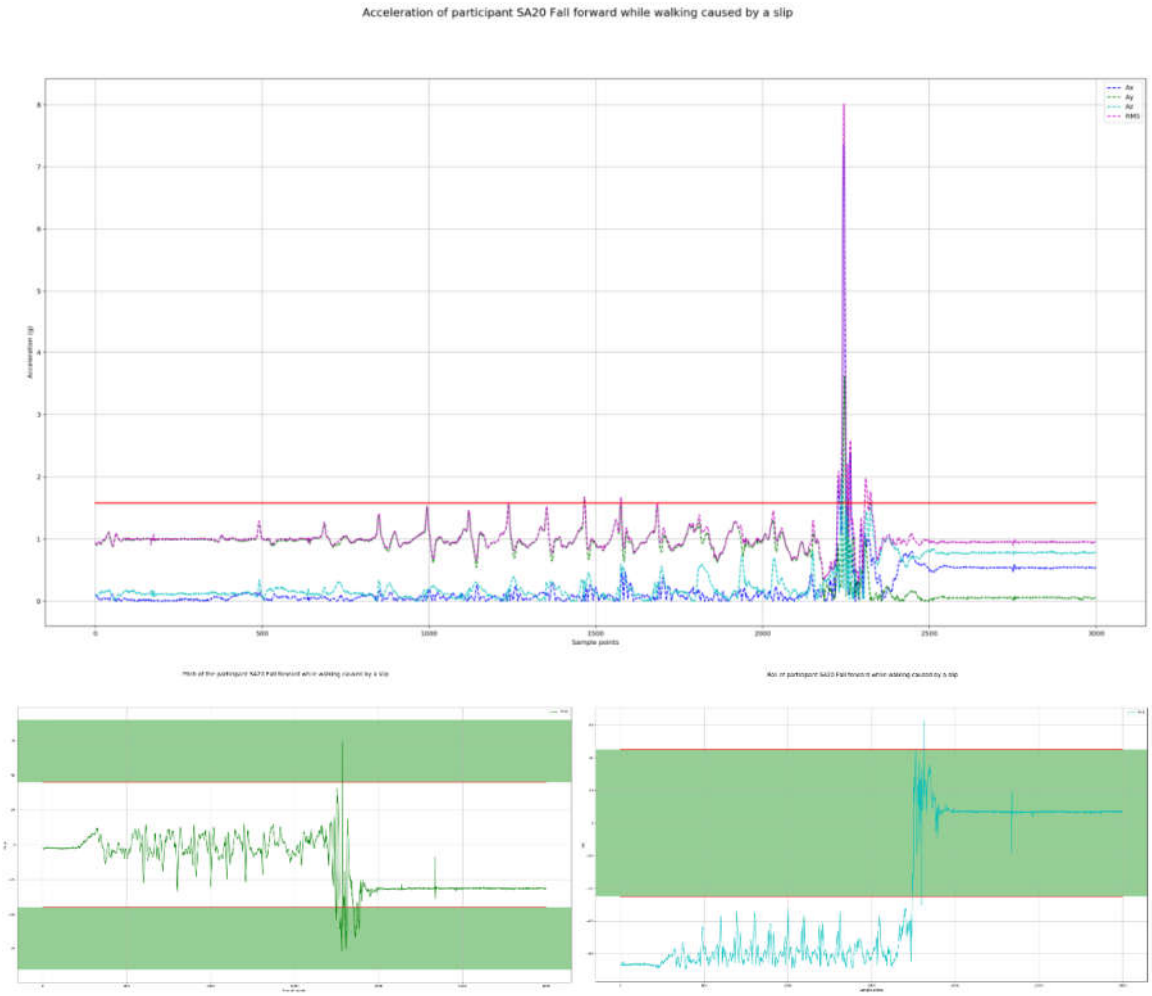


Figure 20 (a-c) Acceleration, RMS, Pitch and Roll value of the adult fall forward while walking caused by a slip. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

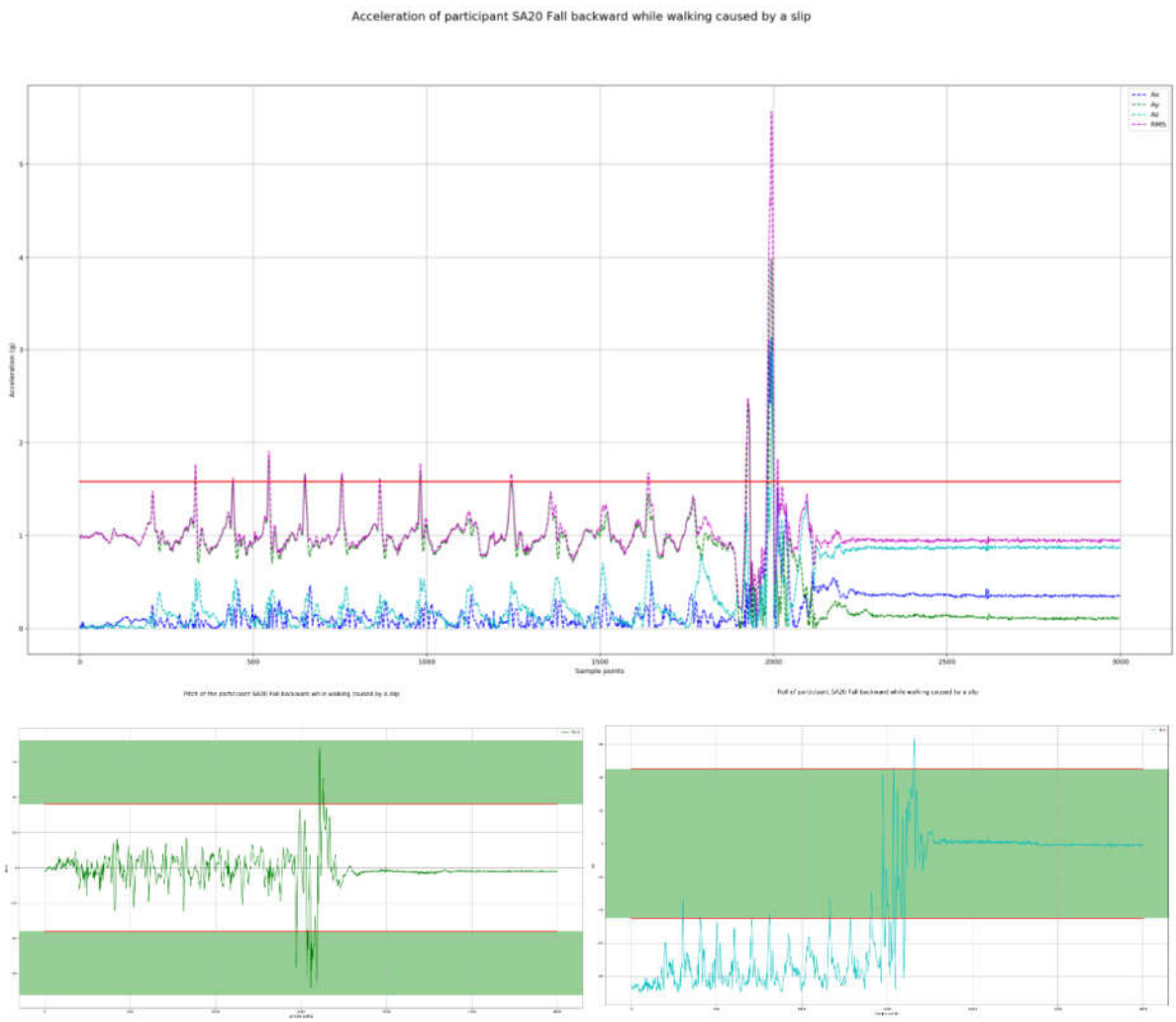


Figure 21 (a-c) Acceleration, RMS, Pitch and Roll value of the adults falling back while walking caused by a slip. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

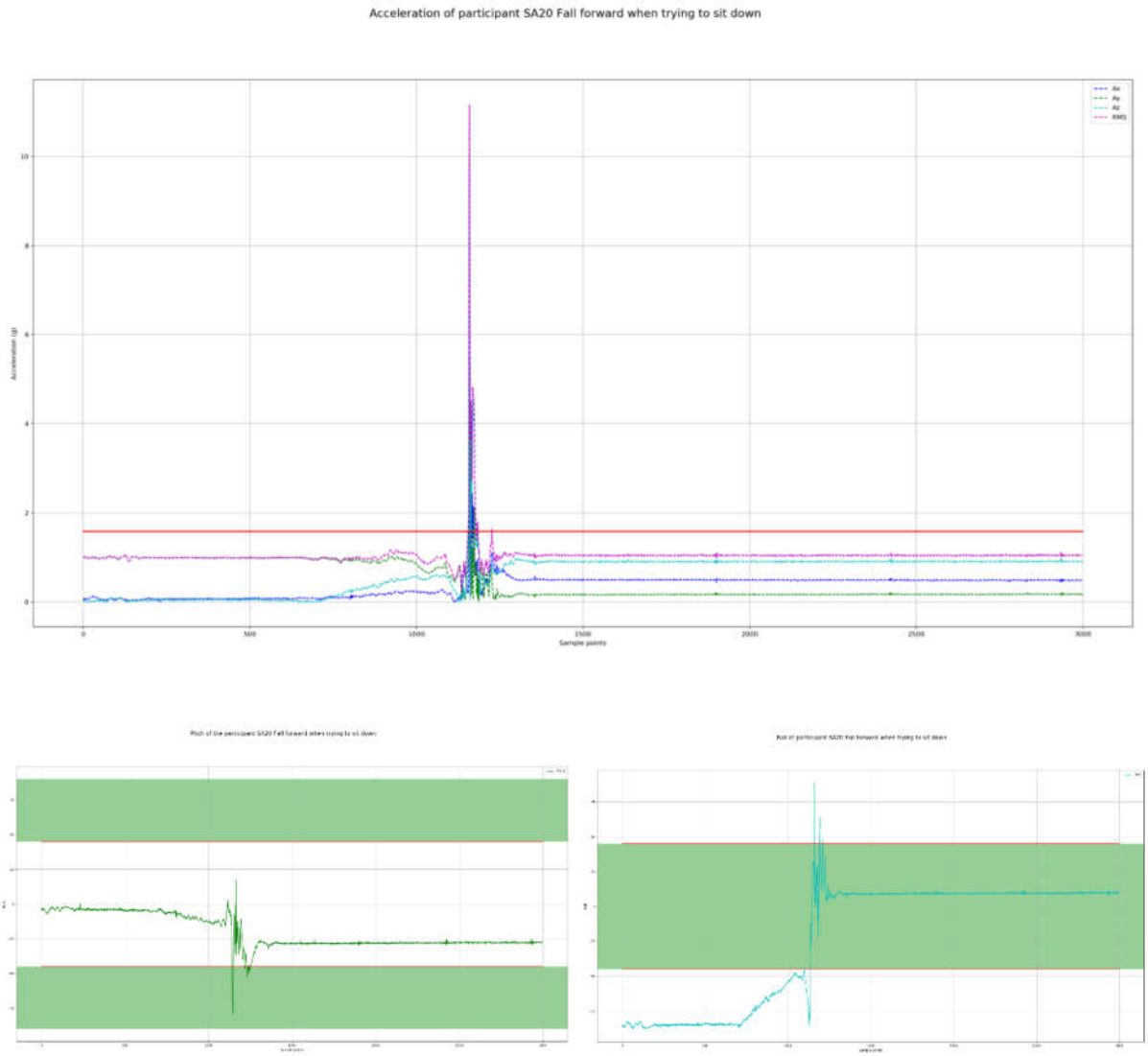


Figure 22 (a-c) Acceleration, RMS, Pitch and Roll value of the adult fall forward when trying to sit down. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

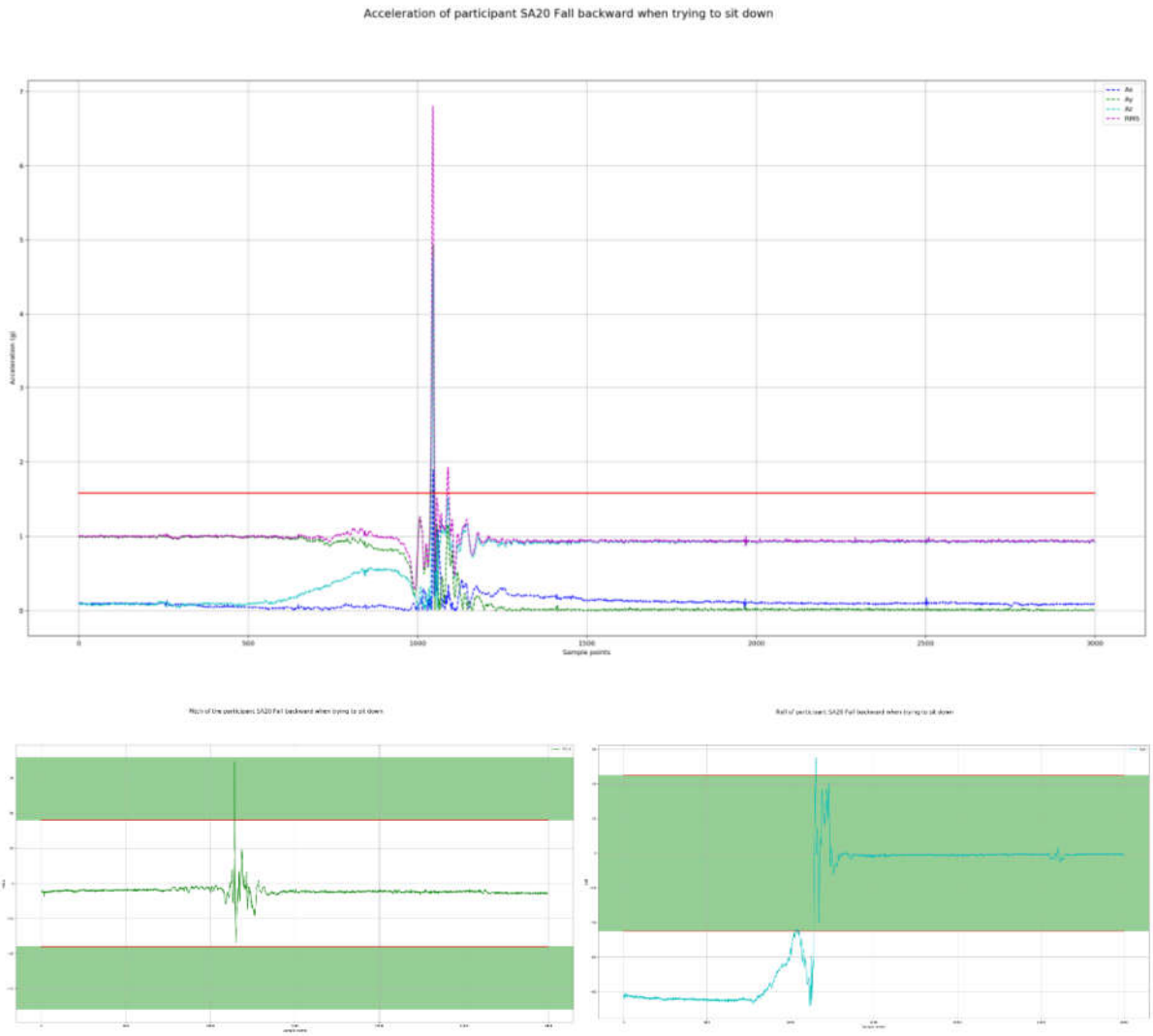


Figure 23(a-c) Acceleration, RMS, Pitch and Roll value of the adult falls backward when trying to sit down. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold areas.

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

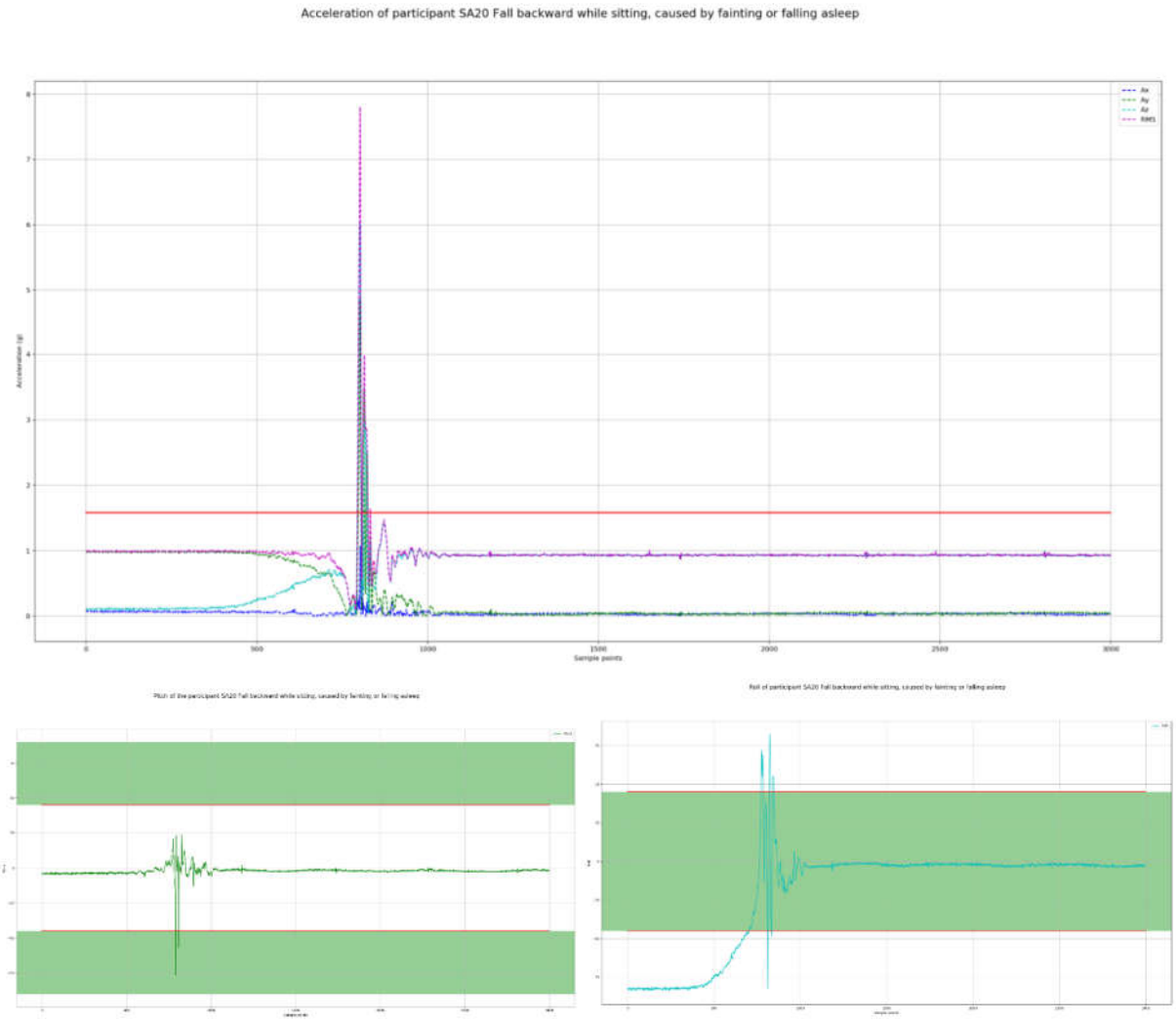


Figure 24 Acceleration, RMS, Pitch and Roll value of the adult fall backward while sitting, caused by fainting or falling asleep. The red line in (a) denotes the RMS threshold, while the green zones in the pitch and roll plots denotes the corresponding threshold

Plot	Labels	Color
a	Ax	Blue
a	Ay	Green
a	Az	Cyan
a	RMS	Magenta
b	Pitch angle	Green
c	Roll angle	Cyan

5.4 Sensor in general context of application

To solve real-world problems, conduct operations in different aspects of our lives and to learn and get values from any physical quantities, sensors are required. Sensors can convert any physical entity (temperature, heat, magnetic field, motion etc.) into electrical signals.

In the context of ambient assisted living, sensors are extremely important. Ambient Assisted Living (AAL) is the area to create smart technology-based solutions for the assistance of people in their day to day lives. Applications in this area includes elderly assistance and monitoring, baby monitoring and nurturing, remote household monitoring and controlling and so on. Interests into this sector are rising significantly in the recent years [11] due to rapid advancement of smart systems and IoT.

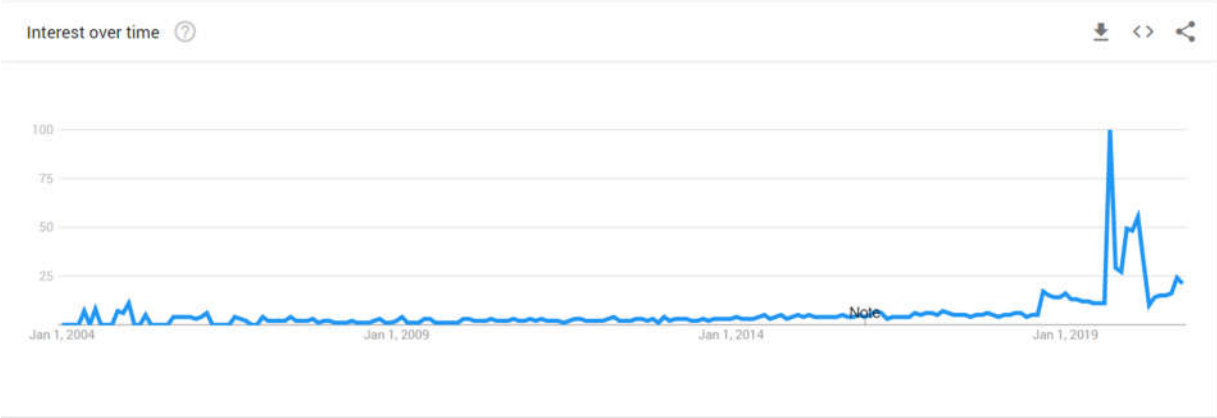


Figure 25 Interest of fall detection over time, from January 2004 to December 31, 2020. The data is taken from Google Trends with the search topic "fall detection." The scale is normalized from 0 to 100 [17]

Elderly fall detection has remained an ever-challenging problem in AAL. Numerous research and solutions have launched in the market to tackle this challenge. Since the motion pattern of our body is inherently dynamic, the complexity of designing any solution is extremely high. Currently, four types of fall detection methodologies are used [18]. They are as follows:

- Wearable sensor-based fall detection
- Visual sensor-based fall detection
- Ambient sensor-based fall detection
- Sensor fusion

Wearable sensor-based fall detection is regarded as one of the key types of sensors and is widely studied, due to the advantages of cost-effectiveness, portability, mobility, and real-time feedback capability [11]. One can detect anomalies in the motion and activities of human being by sensing the acceleration, angular motion, pulse and body pressure via accelerometers, glucometers, gyroscopes, pulse sensors and so on [19, 20].

5.4.1 General requirements of the application

i) Temperature sensitivity

In general, accelerometer (piezoresistive, piezoelectric or capacitive) type sensing for fall detection using threshold-based method depends on the following parameters:

- Frequency response
- Sensitivity
- Acquisition sensitivity
- Noise
- Vibration and external pressure
- Filtering
- Resolution

In the context of piezoresistivity, the following additional parameters are also significant:

- Choice of piezoresistive material (p/n type, Si-crystal orientation etc.)
- Dimensions
- Mode of operations
- Time frame
- Alarm threshold

Some important parameter considerations for our design are given below:

i. **Noise handling:** Since the range of our measurement of acceleration is lower than other contexts and the dynamic factor of humans are significantly high, a lot of noise or corrupted data can be found during measurement. Thus, improving the resolution and maintain the stable margin is significant for our process. Factors such as EMF, mechanical vibrations, material issue can be significant.
   Therefore, our design also includes a filtration subsystem that will filter out the external noise components. Internal noises can be mitigated during the sensor design and fabrication part.
ii. **Sensitivity:** The sensitivity of an accelerometer is defined as the ratio of the electrical out to the mechanical out. Depending on the choice of unit, it is either expressed by pC/g

(pico coulombs-per-g) or mV/g (millivolts-per-g). Since piezoresistive accelerometers offer low sensitivity, external amplifying sub-system is required [21]. In the case of fall detection, we are interested in the sensitivity across all the axes of the sensor. Therefore, for each axes the measurement of transverse versus axial sensitivity is important, and their tolerances should be limited between 3% to 5%.

iii. **Acquisition sensitivity:** As already discussed, acquisition sensitivity is extremely significant to acquire and produce response in near real-time, since our problem deals with the factor of human health. This sensitivity depends on the frequency response, system clock and power supply to the device.

iv. **Mode of operations:** Since using accelerometers fall needs to be detected from multiple postures and trajectories, it is necessary to learn the mode of operation of the designed solution. Through this parameter, users will be able to know the real-time detection limitations when the device will be used.

v. **Time frame of fall:** In threshold-based designs, time frame is an important parameter choice to be considered by the designers, as well as for the user. Through this parameter, users will be able to know the detection speed of falls, and swiftly an SOS call is going to be launched to the concerned people. As described above, this time frame has to be decided by the designer via observation or using advanced methodologies, such as machine learning.

vi. **Cost:** Accelerometer sensors suitable for fall detection are widely available now a days. Accelerometer-based fall detection solutions are becoming popular among the elderly people due to the cost-effectiveness. Prices of such solutions for general people ranges from $88 to $330 in the market [9, 10, 22]. Products and startups such as Wellnest, Tango Belt, Hip'Safe, Smart Caregiver are some of the widely popular products leading today's market [22].

5.4.2 Smart sensor in a more complex system

The system implementation includes acceleration sensing, signal condition and decision making which is shown in the figure below. When the person is experiencing fall, there is a sudden change in acceleration which can be detected by the piezoresistive sensor through change in resistance ΔR. Using the op-amp, the change is resistance is converted into voltage and then amplified as the converted voltage is very less for the VCO. VCO converts this amplified voltage into frequency which is then fed to the digital system.

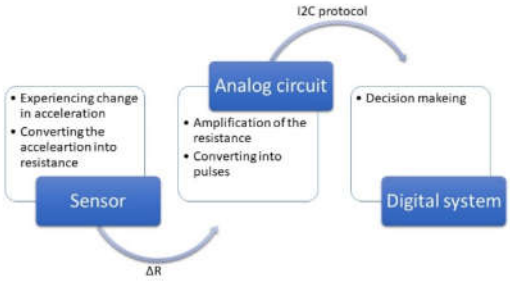


Figure 26: System implementation for fall detection

For the communication between analog front end and digital system, I2C protocol is used. I2C is a serial communication protocol therefore the data is transferred bit by along a single wire. Also the I2C is synchronous which results in the synchronized output of bit. The micro-controller uses the algorithm to decide based on the sudden change in acceleration. If the acceleration is accounted higher than the threshold acceleration, then the fall is detected which sends an alert in the form of an alarm.

5.5 Model Specifications

5.5.1 Theoretical Background

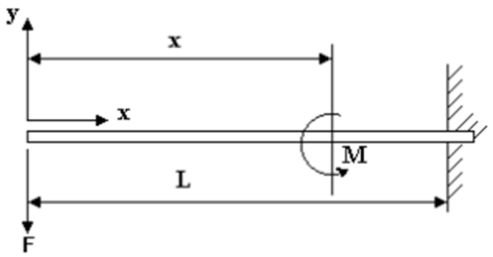


Figure 27: Cantilever loaded with point load

For a beam loaded with a force F at its end, as shown in figure 3.1, the moment is given by [23]:

$$EI\frac{d^2y}{dx^2} = -Fx \quad (5)$$

Integrating w.r.t x and we get

$$EI\frac{dy}{dx} = \frac{-Fx^2}{2} + A \quad (6)$$

Integrating again and we get

$$EIy = \frac{-Fx^3}{6} + Ax + B \quad (7)$$

Applying the boundary conditions:

At x=L, y=0 (no deflection)

At x=L,

$$\frac{dy}{dx} = 0$$

Substituting x=L and $\frac{dy}{dx} = 0$ in equation (6). This gives

$$EI\frac{dy}{dx} = \frac{-FL^2}{2} + A$$

hence $A = \frac{FL^2}{2}$

Substituting $A = \frac{FL^2}{2}$, y=0 and x=L into equation (7) and the equations become

$$EI\frac{dy}{dx} = \frac{-Fx^2}{2} + \frac{FL^2}{2} \quad (8)$$

$$EIy = \frac{-Fx^3}{6} + \frac{FL^2x}{2} - \frac{FL^3}{3} \quad (9)$$

Since we need the slope and deflection at the end of the free end where x=0, we must substitute x=0 into equation (8) and (9) equations

Slope at free end:

$$\frac{dy}{dx} = \frac{FL^2}{2EI} \tag{10}$$

Maximum deflection at free end

$$y = -\frac{FL^3}{3EI} \tag{11}$$

Here, the spring constant, *k*, of the bending beam can be found as

$$F=\frac{3EI}{L^3}(-y)$$

$$\Rightarrow\ k = \frac{3EI}{L^3} \tag{12}$$

Spring compliance is the inverse of the spring constant; $S=\frac{3EI}{L^3}$, where E is the Young modulus of the cantilever material, and I is the second moment. Here, for the rectangular cross-section of the beam,

$$I = \frac{ab^3}{12} \tag{13}$$

where a and b is the thickness and deflection y<<L

Another important parameter is the maximum strain under a given load. The beam experiences maximum tensile stress at the fixed top end of the beam. Hence, maximum tensile strain is also experienced at the place of maximum stress. This value of maximum strain, $\varepsilon_{\max}$, is given by [23]:

$$\varepsilon_{\max} = \frac{Lb}{2EI}F \tag{14}$$

Using equation 12 in 14

$$\varepsilon_{\max} = \frac{3by}{2l^2} \tag{15}$$

When a piezoresistor is placed at this zone of maximum stress, the piezoresistor undergoes the same stress. This applied mechanical stress induces a proportional alteration of material resistivity in the piezoresistor.

If a relatively long and narrow resistor is defined in a planar structure (where thickness is in order of half a micron), then the primary current density and electric field are both along the long axis of the resistor, which do not coincide with the cubic crystal axes. This situation helps the piezoresistive coefficients, which are derived from the field-current relationship, to be simplified as [24]:

$$\frac{\Delta R}{R} = \pi_l\sigma_l + \pi_t\sigma_t \tag{16}$$

Where R is the resistance of the resistor, and the subscripts *l* and *r* refer to longitudinal and transverse refer to stresses along the resistor axes. These coefficients in silicon depend on the crystal orientation and the dopant type as listed in Table 8:

Table 8: Room- temperature coefficients for n-type and p-type silicon

	$\pi_{\parallel}$ $[1\cdot10^{-11}\,\mathrm{m^2/N}]$	$\pi_{\perp}$ $[1\cdot10^{-11}\,\mathrm{m^2/N}]$	
p-type	7	-1	In < 100 > direction
	72	-66	In < 110 > direction
n-type	-102	53	In < 100 > direction
	-31	-18	In < 110 > direction

From the spring-mass-damper representation of a piezoresistive cantilever accelerometer, the undamped resonance frequency, or the natural resonance frequency, can be derived as [25]:

$$\omega_n = \sqrt{\frac{k}{m}} \tag{17}$$

$$\Rightarrow f = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \qquad where\ \ 2\pi f = \omega_n$$

Spring compliance $k = {}^{1}/_{S\,.\,m}, from\ eqn\ (12)$

$$f = \frac{1}{2\pi}\sqrt{\frac{1}{S \times m}} \tag{18}$$

A LabVIEW GUI was created as an optimization tool that achieved the resistive change between 0.2-0.3% as a function of geometrical dimensions. Figure 28 shows the LabVIEW GUI while Figure 29 displays the block diagram of the code

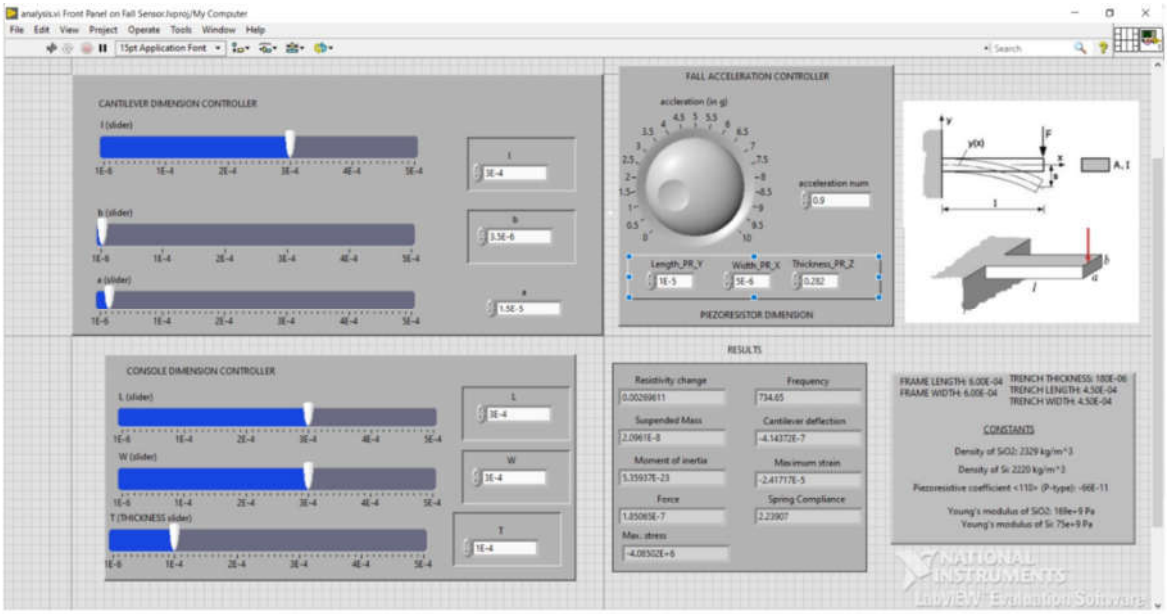


Figure 28: LabVIEW GUI to Optimise Calculations

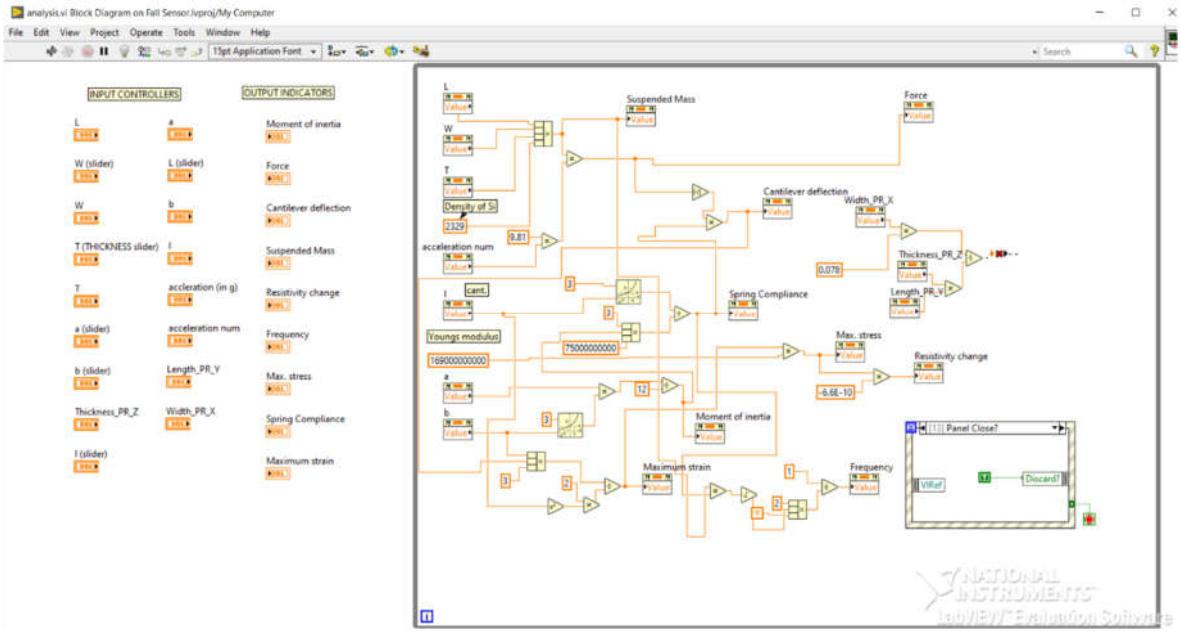


Figure 29: LabVIEW Block Diagram of code

s

Structure	Length (x) (µm)	Width (y) (µm)	Thickness (z) (µm)
Cantilever (neck)	300	15	3.5
Console Mass	300	300	100
Piezoresistor	5	10	0.282
Trench	650	650	120
Model Domain	1000	1000	120

With the values from Table 9, the accelerometer was designed in Ansys Design Modeler.

Table 10: Constants for theoretical calculations

Acceleration threshold	0.9g (g=9.81 m/s^2)
SiO_2 Density	2220 kg/m^3
p-type Si Density	2329 kg/m^3
Young Modulus (Si) <110>	169 Gpa
Young Modulus (SiO_2) <110>	75 GPa
Piezoresistive coeffficient	-66E-11 m^2/N

Here the Piezoresistive coefficient was chosen as -66E-11 m^2/N since the location of the piezoresistor is transversal to that of the cantilever [24]. -66E-11 m^2/N is the transverse piezoresistive coefficient. To calculate the resistance change in equation (16), we ignore π_l, since the stress experienced by resistor will be in the transverse direction to its axis. Hence equation (16) reduces to $\frac{\Delta R}{R} = \pi_t \sigma_t$, where

$\sigma_t = E_{s_i} \varepsilon_{\max}$

$$\Rightarrow \frac{\Delta R}{R} = \pi_t E_{s_i} \varepsilon_{\max} \quad (19)$$

$$\Rightarrow \frac{\Delta R}{R} = \pi_t E_{s_i} \frac{3by}{2l^2}$$

This is the stress experienced by the resistor in the resistor that induces a resistance change due to the piezoelectric effect.
Figure 30 displays how the piezoresistive coefficients must be chosen to determine the stress and strain in resistor according its placement with respect to the cantilever

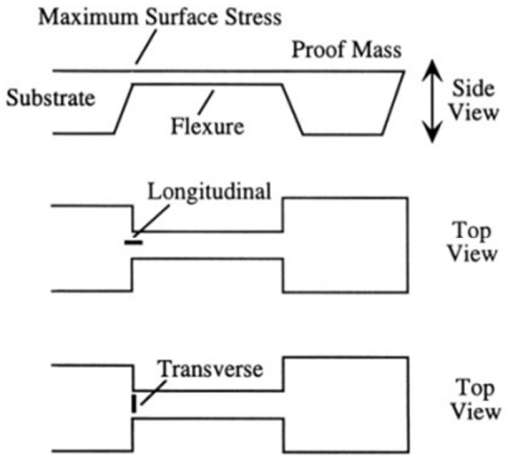


Figure 30: Lateral and transverse piezoresistor placements in a bending cantilever

Table 11 displays the calculated values from LabVIEW GUI.

Table 11: Calculated values from theoretical analysis (LabVIEW simulation)

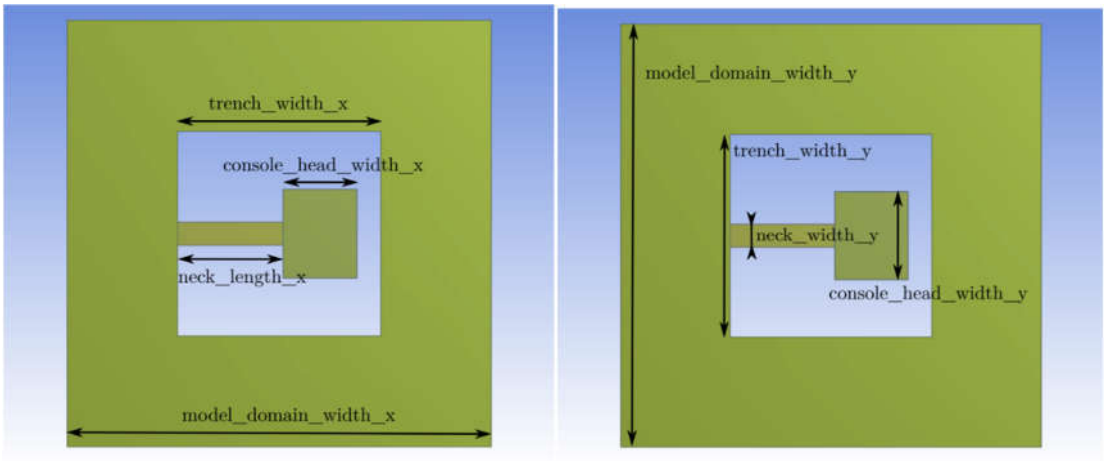
Calculated value	Vaule	Unit
Force	1.85065E-07	N (kg.m/s^2)
Pressure	2.128	Pa
Deflection of cantilever [s]	4.14E-07	M
Strain	2.417E-05	
Stress Parallel	4.085E-06	Pa
Resonant Frequency	734.65	Hz
Spring compliance [S]	2.239	N/m
Moment of Inertia [neck]= ab^3/12	5.534E-27	Kg.m^2

$\frac{\Delta\rho}{\rho}$	0.269	%

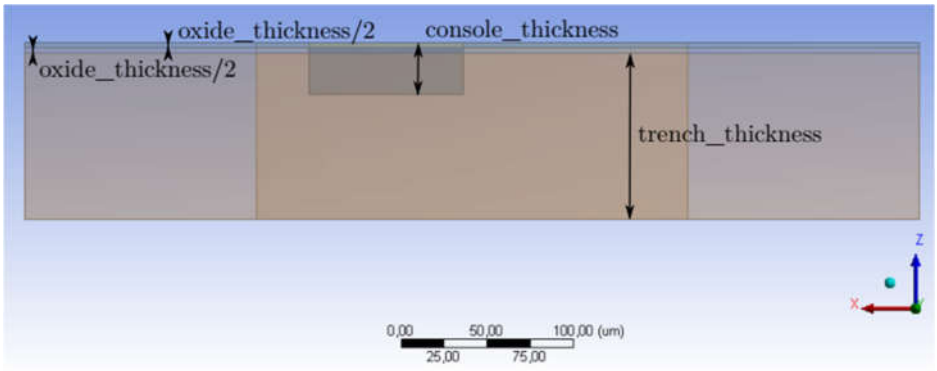
5.6 ANSYS Model preparation

5.6.1 Geometry for the model

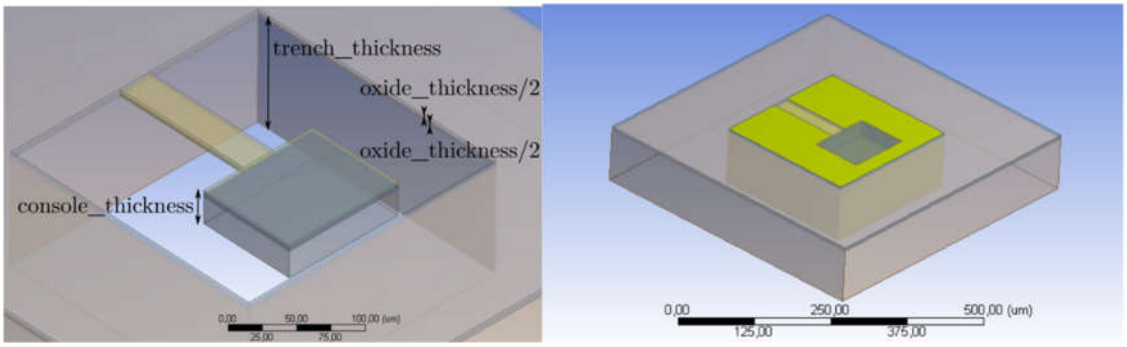
As mentioned in previous sections, three accelerometers will be positioned so as to detect the acceleration changes along the x, y and z axis with respect to a person's posture. However, the geometry for all three accelerometers will be the same since the acceleration threshold for a fall event along all three axes are the same, i.e. 0.9g.
The geometry of the accelerometer is a typical cantilever neck attached to a silicon body. The body is a 1000 µm X 1000 µm X 120 µm rectangular body with a trench of 650 µm X 650 µm etched in the middle. Attached to the cantilever tip, is a console-head of 300 µm X 300 µm X 100 µm volume. A piezoresistor of dimensions 5 µm X 10 µm X 0.282 µm is placed at the cantilever base. When the console-head undergoes a force due to acceleration, the force on the tip of the cantilever due to the console-head can be approximated as a transversally loaded beam cantilever problem.
The geometry was designed through a set of parametric values to simplify the task of geometry generation and subsequent simulation. The defined parameters are shown in Figure 31.



(a)



(b)



(c)

Figure 31: (a) Parameters in X-Y plane (b) and (c) Parameters in X-Z Plane

The bulk was made of silicon. A thin layer of silicon dioxide SiO_2 surface (oxide thickness) covers the top of the console-head, the neck and the frame. A square trench of Silicon is etched in the body. The neutral plane of the neck and head is parallel to the oxide-surface.

5.6.2 Meshing

For meshing, a dominant hex method was employed for the console, resistor and the neck, with the entailing default values. Mesh element size of 2μm was chosen for the resistor element while element size of 5 μm was chosen for the neck and console. For the silicon bulk, the element size was chosen automatically by Ansys. Element sizes were chosen smaller for neck, resistor and console regions to capture as many variations as possible. Meshing was conducted in the order

i. Console 1, 2 and 3 (includes console-head and SiO_2 layers)
ii. Resistor
iii. Neck 1 and 2
iv. Remaining domains

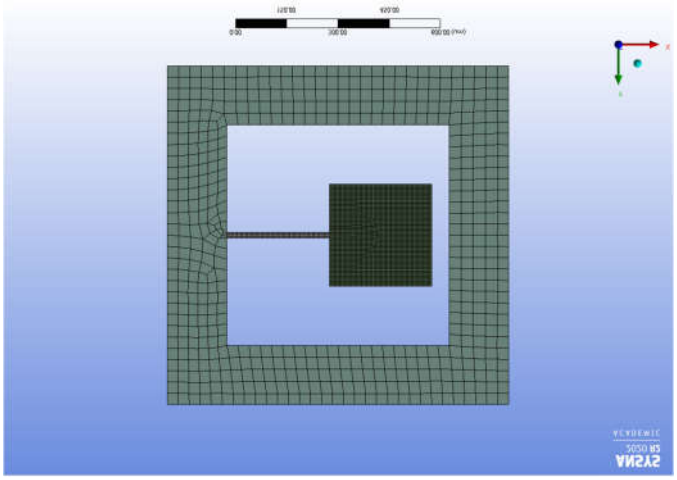
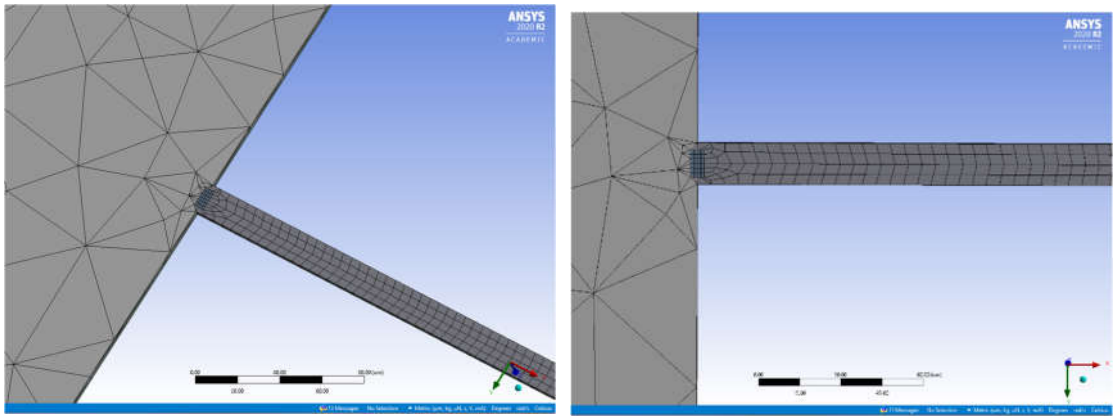


Figure 32: Meshing for whole structure top View (Air is hidden)

Figure 33 shows the detailed meshing in the piezoresistor region.



(a) (b)

Figure 33: Meshing in piesoresistor region

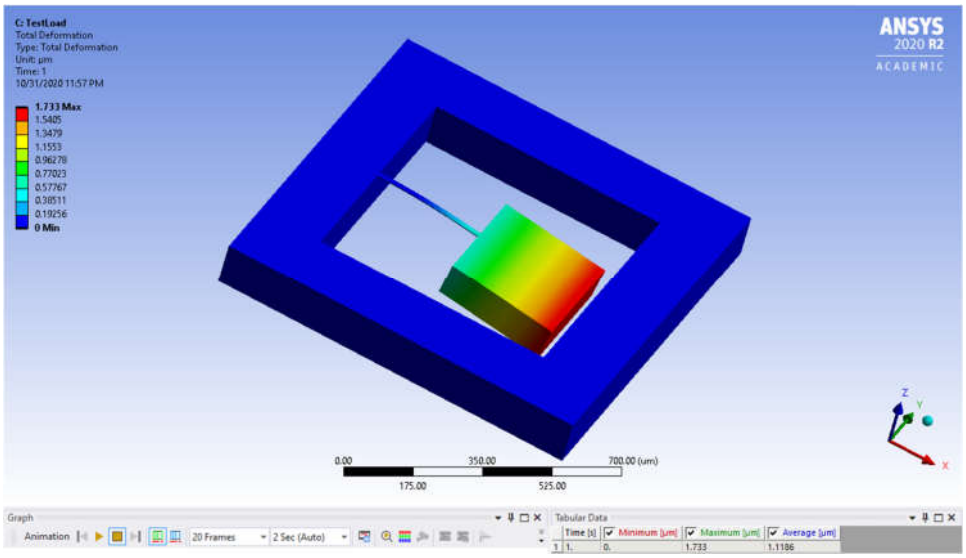
5.6.3 Test Load analysis

The Test Load analysis required a pressure value that corresponds to the acceleration of 0.9g on the console-head. This was calculated as

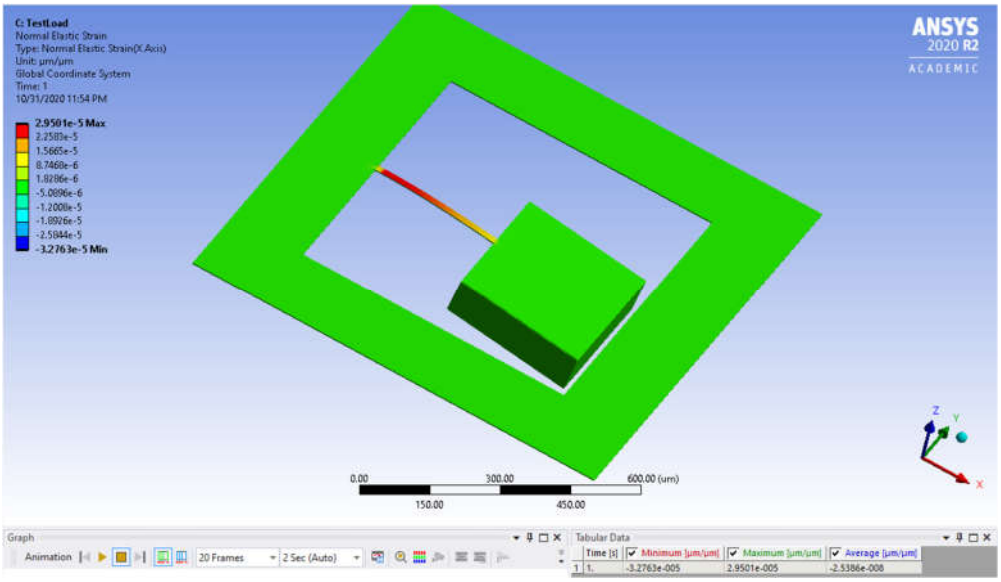
$F = m_{console} a$

$P = \frac{F}{A_{Console}}$, where $A_{Console}$ is the surface area of the console along the x-y plane

Pressure was calculated as 2.128 Pa. This is equivalent to the fall impact acceleration of 0.9g. The cantilever undergoes a a total deformation of 1.733 µm as shown in the figure 34. Figure 35 shows the corresponding strain from the fall event.



. Figure 34: Total Deformation under test load of Pressure 2.128 Pa



. Figure 35: Total strain under test load

5.6.4 Comparison with theoretical calculations

Table 13: Comparison of Theoretical and Simulated results of structure under test load

	Theoretical	Simulated
Deflection	4.143 um	1.733 um
Strain	2.417E-05	2.9501E-05

5.6.5 Element load analysis

In the element load analysis a pressure of 1kPa and acceleration due to gravity, g, was applied in the z-direction (perpendicular to console-head). Application of accelerations caused a maximum displacement of 1.8424 µm a shown in Figure 36. This means that gravity can have considerable effect on the sensor performance.

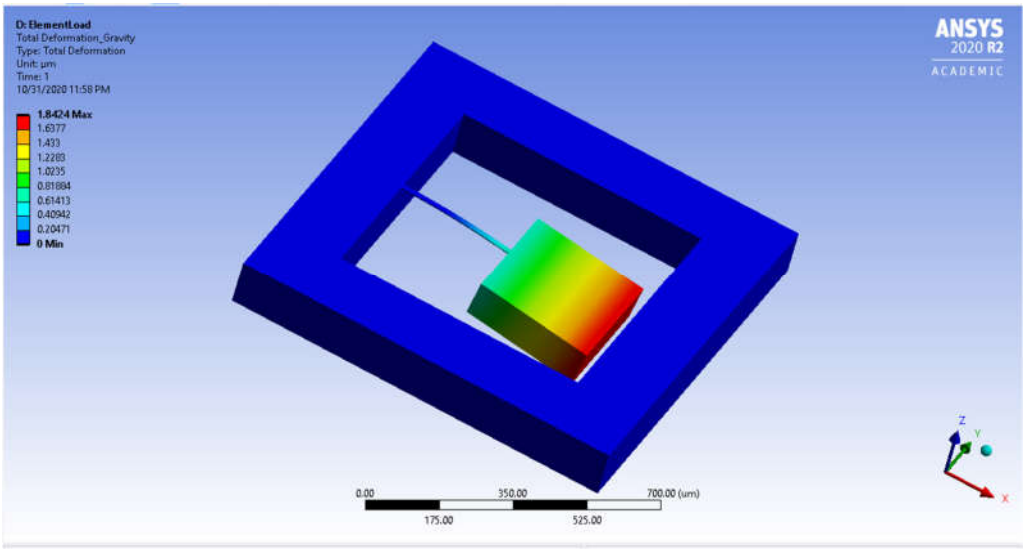


Figure 36: Total Deformation under Element load (Acceleration =g)

Figure 37 shows deformation due to application of 1kPa Pressure.

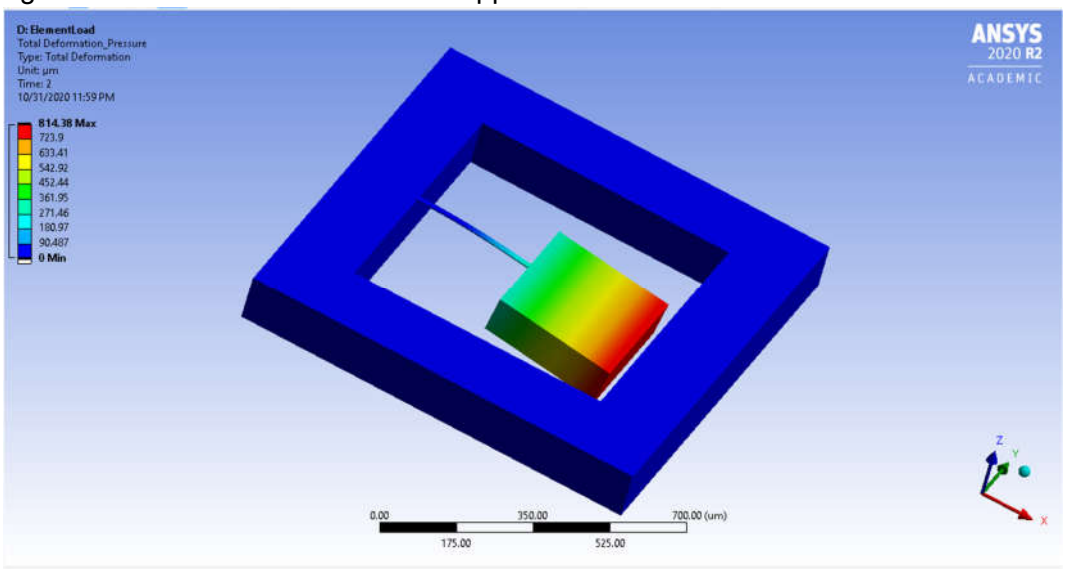


Figure 37: Total Deformation under Element load (Pressure = 1Kpa)

Results from Test and Element Analysis

From the Test and Element Load analysis, it is seen that the results when the threshold acceleration of 0.9g and the gravitational load of 1g produced almost

similar deflections. This indicates that gravity can have a significant effect on the device performance. A solution for this will be later proposed in the project.

Table 14: Comparison of results of Test Load and Element Load.

	Test Load	Element Load
Deflection	1.733 um	1.8424 um

5.7 ROM of the MEMS device

Next Modal analysis was performed after choosing relevant master nodes. The procedure is detailed below.

5.7.1 Modal analysis with the modal contribution factors

Modal Analysis was performed for 9 modes, the details of which are obtained from Solution Information as shown below:

```
TOTAL NUMBER OF MODES DEFINED FOR ROM TOOL =  9

 MODE  1
  MODE ID          =  1
  RELEVANCY        = DOMINANT
  FREQUENCY        =   429.43
  DAMPING RATIO    =   0.0000
  LOWER BOUND DISPL. =  -14.706
  UPPER BOUND DISPL. =   14.706
  MODAL SCALE FACTOR =  0.68000E-01
  NUMBER OF STEPS IN FIT RANGE =  11
  LAST AUTOMATED SELECTION PROCEDURE DETERMINED
  A MODAL CONTRIBUTION FACTOR OF    99.571     PERCENT

 MODE  2
  MODE ID          =  2
  RELEVANCY        = UNUSED

 MODE  3
  MODE ID          =  3
  RELEVANCY        = UNUSED

 MODE  4
  MODE ID          =  4
  RELEVANCY        = DOMINANT
  FREQUENCY        =   5012.8
  DAMPING RATIO    =   0.0000
  LOWER BOUND DISPL. =  -0.29412
  UPPER BOUND DISPL. =   0.29412
  MODAL SCALE FACTOR =   3.4000
  NUMBER OF STEPS IN FIT RANGE =  5
  LAST AUTOMATED SELECTION PROCEDURE DETERMINED
  A MODAL CONTRIBUTION FACTOR OF   0.36863     PERCENT

 MODE  5
  MODE ID          =  5
  RELEVANCY        = UNUSED
```

```
MODE  6
 MODE ID          =  6
 RELEVANCY          = UNUSED

MODE  7
 MODE ID          =  7
 RELEVANCY          = UNUSED

MODE  8
 MODE ID          =  8
 RELEVANCY          = UNUSED

MODE  9
 MODE ID          =  9
 RELEVANCY          = UNUSED
```

From the above details it can be concluded that Mode 1 contributes the most to the device performance with a contribution factor of 99.571%. Mode 4 contributes too, but only with 0.36863 %. Rest of the modes are unused and can be discarded for future steps. Hence, for ROM model generation Mode 1 and 4 are chosen. Figure 37 and Figure 38 shows deformation due to Mode 1 and 4 respectively.

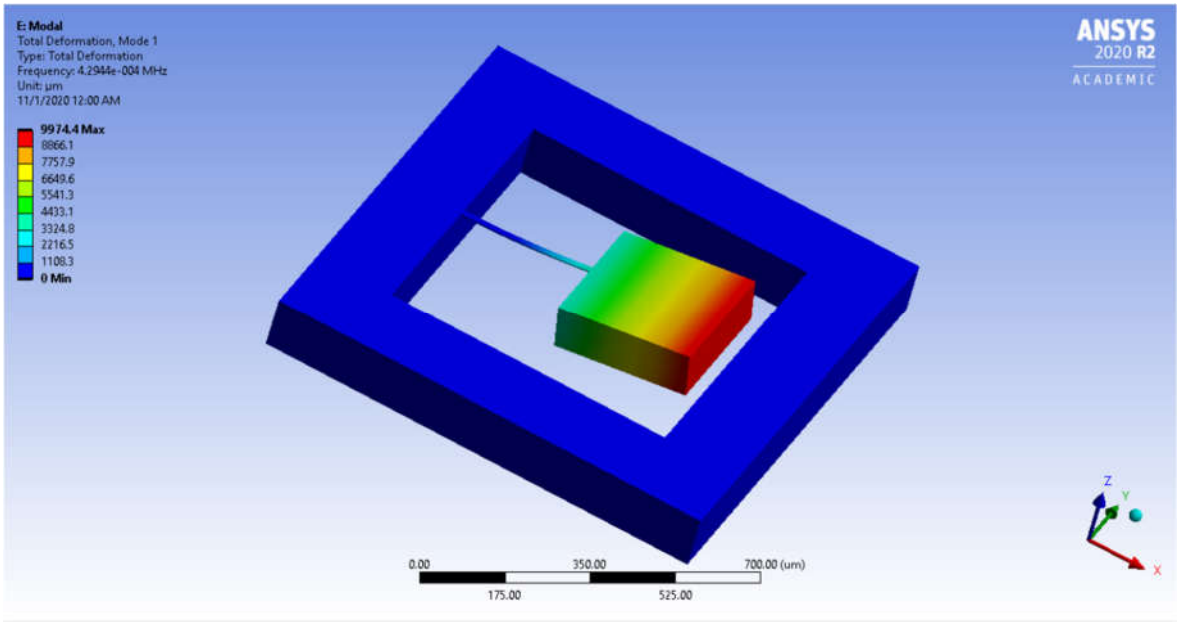


Figure 38: Structure movement in Mode 1

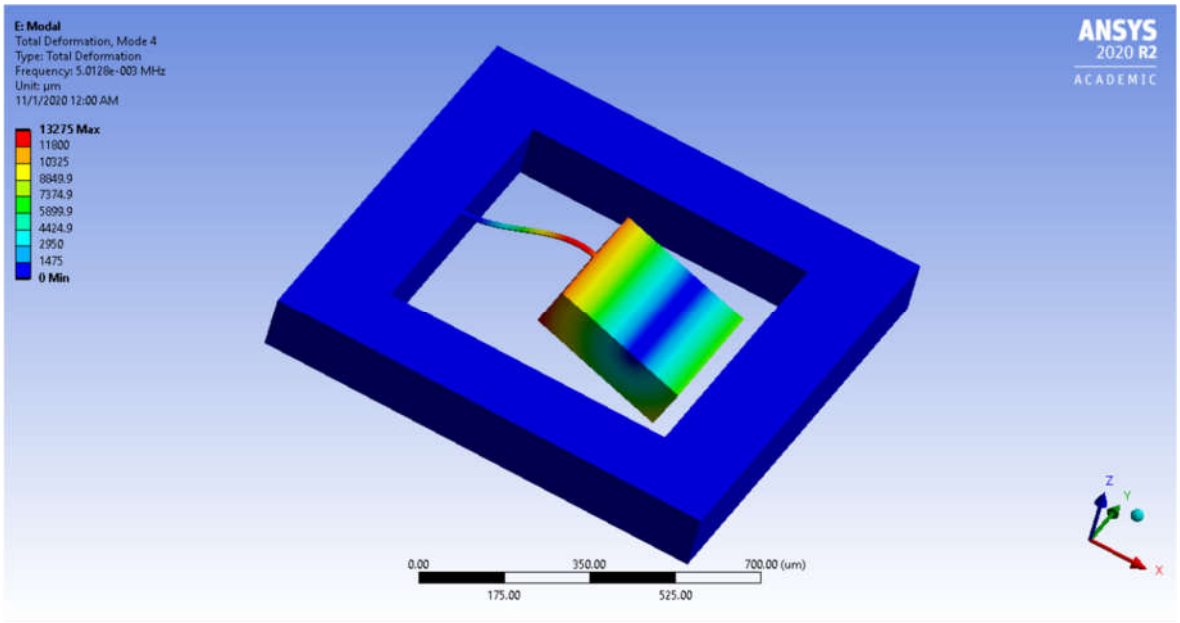


Figure 39: Structure movement in Mode 4

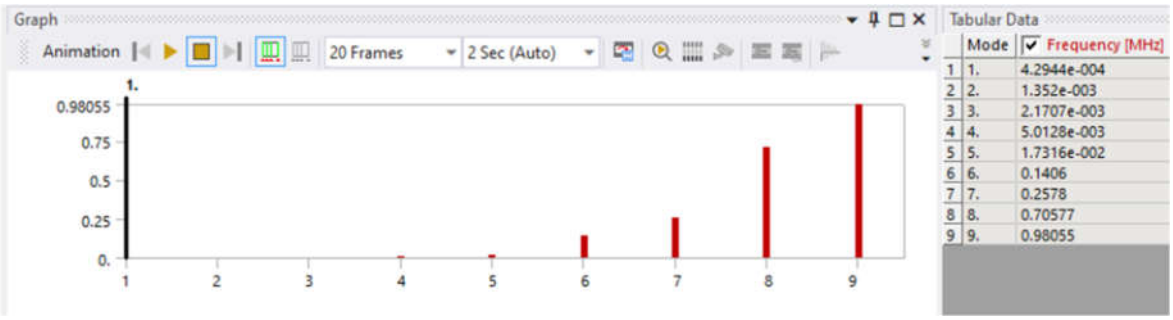


Figure 40: Contribution of Modes to Structure movement

Next, Mode 1 and 4 are chosen as the dominant modes in the ModeSelect script, as shown in Figure 10. Mode 1 is set as DOMINANT while Mode 4 is set as RELEVANT due to their obvious contribution reasons.

```
RMMSELECT,3,'tmod',-15,15
RMMLIST

RMMRANGE,1,'DOMINANT',,,6,0.05      !use 6 steps for mode 1
RMMRANGE,4,'RELEVANT',,,5,0.05      !use 5 steps for mode 4

RMSAVE,file,rom                  !Save ROM database
```

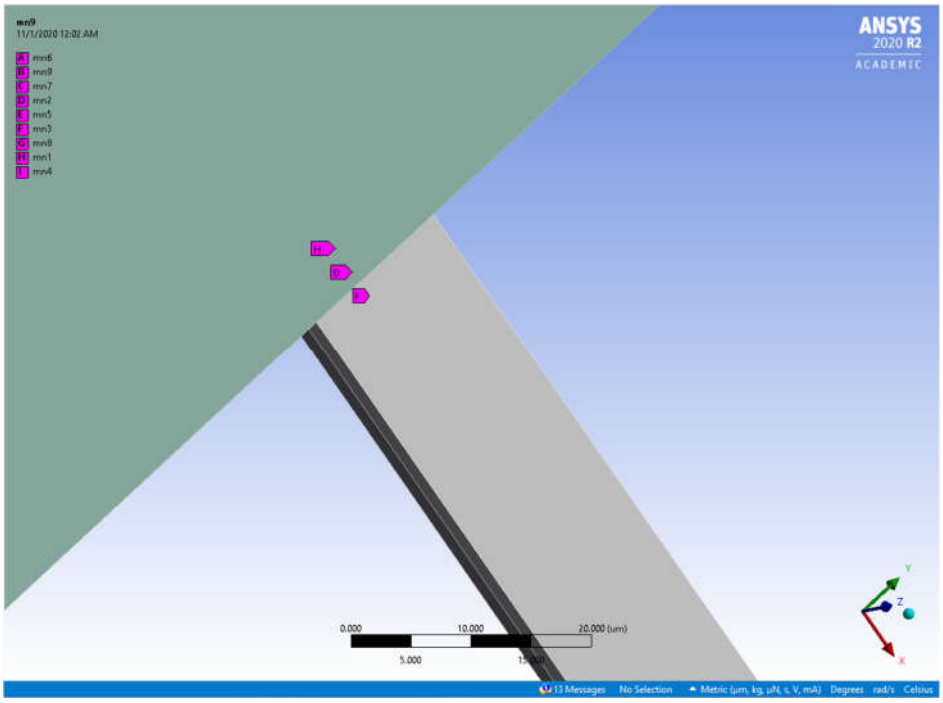
Figure 41: Choosing Modes for ROM generation

5.7.2 Defining relevant master nodes of the investigated structure

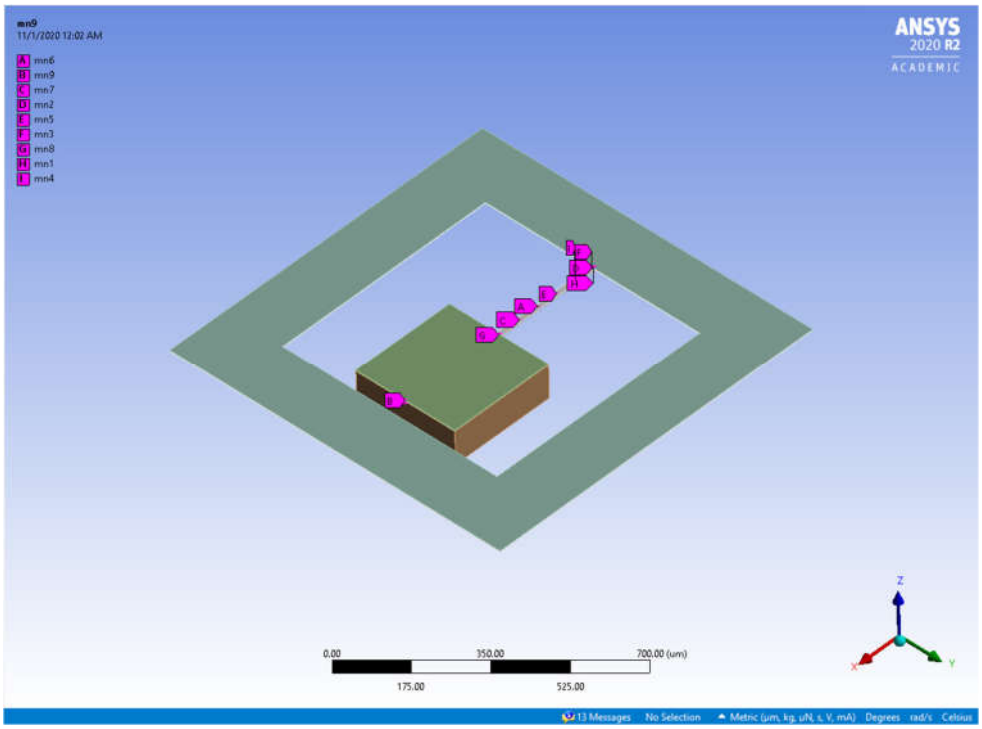
A total of 9 master nodes have been selected for the next section of ROM Analysis (9 is the maximum limit of nodes that can be selected).

- Three nodes were selected along the resistor mid-axis
- One node was chosen at the tip of console
- One node (Node G- mn8) was chosen at tip of cantilever, near the neck-console junction
- Rest of the nodes were chosen along the mid-line of the neck

Figure 42 shows the selection of nodes on the structure.



(a)



(b)

Figure 41: Placement of Master Nodes on the structure

5.7.3 ROM generation

ROM Model was successfully generated with the following element nodes

- Pressure- 1kPa
- Acceleration- 9.81 m/s^2

The files, file.rom and file_104.pcs were generated in the solver directory. These were used for the SSIT simulation subsequently. Besides these files, other generated files were used too, to obtain graphic representation of some of the analysis on master nodes through ANSYS Mechanical APDL. Apdl scripts were written for the following simualtions in the next section.

5.8 Node and displacement analysis

5.8.1 Moving margins of the device

A typical fall event occurs when the device undergoes an acceleration of 0.9g. In this section, the accelerometer performance is analysed under a typical fall event.
From Figure 43, it is seen that the maximum deflection of the cantilever and proof mass is contained within the trench thickness. Also, there is enough space between the Silicon body and the console-head in the trench that prevents their contact during the fall event. Thus, it can be concluded that the device will perform as expected during the fall.

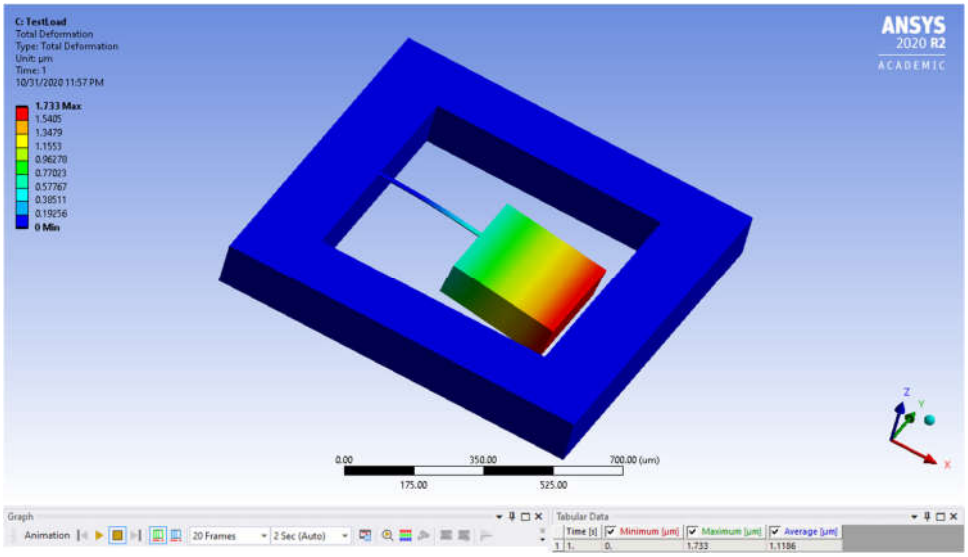
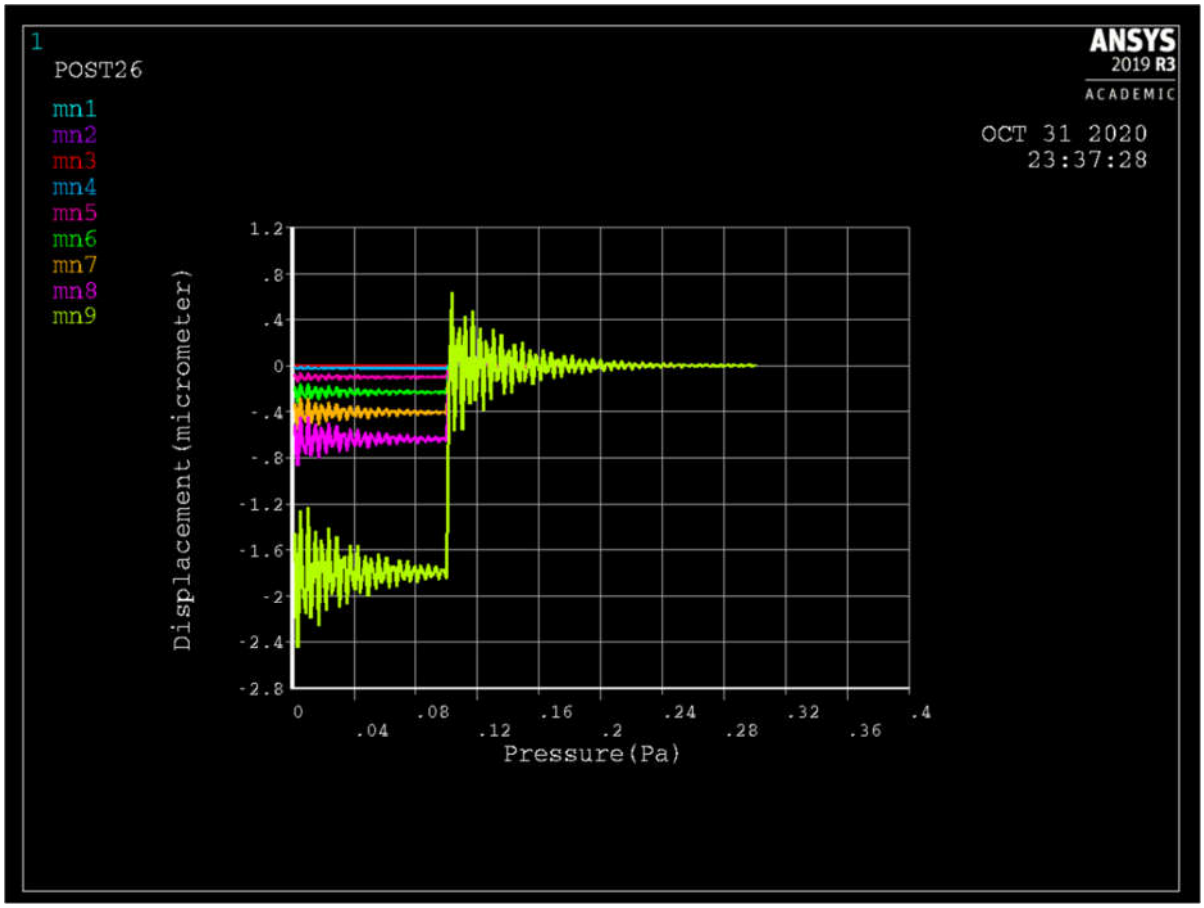


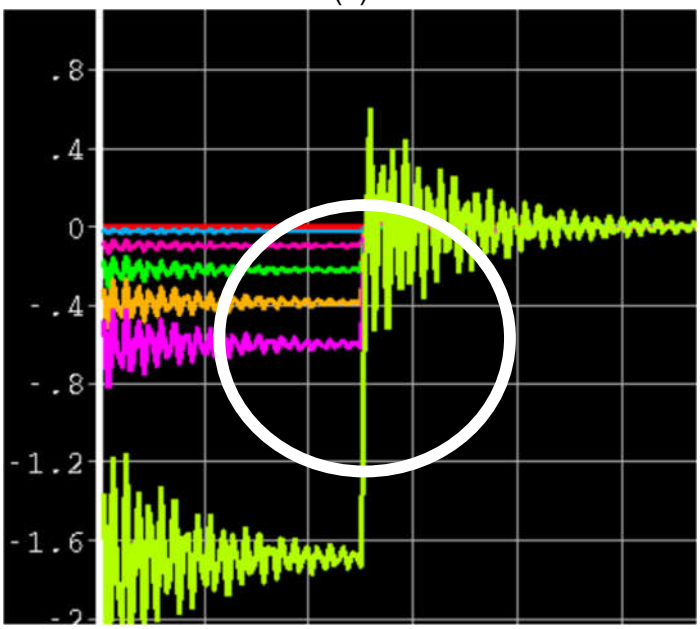
Figure 42: Deformation of Structure under 2.128 Pa Load (0.9g acceleration load)

5.8.2 Master node displacement vs pressure analysis

Once the ROM generation was completed, the impulse response of the device was plotted in Ansys Mechanical to graphically understand the significance of choosing node mn8 (node at neck-console junction) as the main node to calculate strain in piezoresistor.
From figure 44, it is seen that maximum displacement is provided by mn9. This can be ignored because mn9 is the node at the tip of the console-head, which is of no importance to us. Beneath the lime green graph of mn9, the pink graph of mn8 is visible. Here, the device was subjected to a pressure load of 2.128 Pa.



(a)



(b)

Figure 43: Master Node Displacements for acceleration ramp

Acceleration was applied as ramp to obtain the graph in Figure 44. The input ramp was defined as a vector from 0 to 2g (approx.. 20 m/s^2). The master node dispalcements under the acceleration ramp input was plotted here. Displacements are negative since the force applied is in the negative z-direction (on top of the console head).

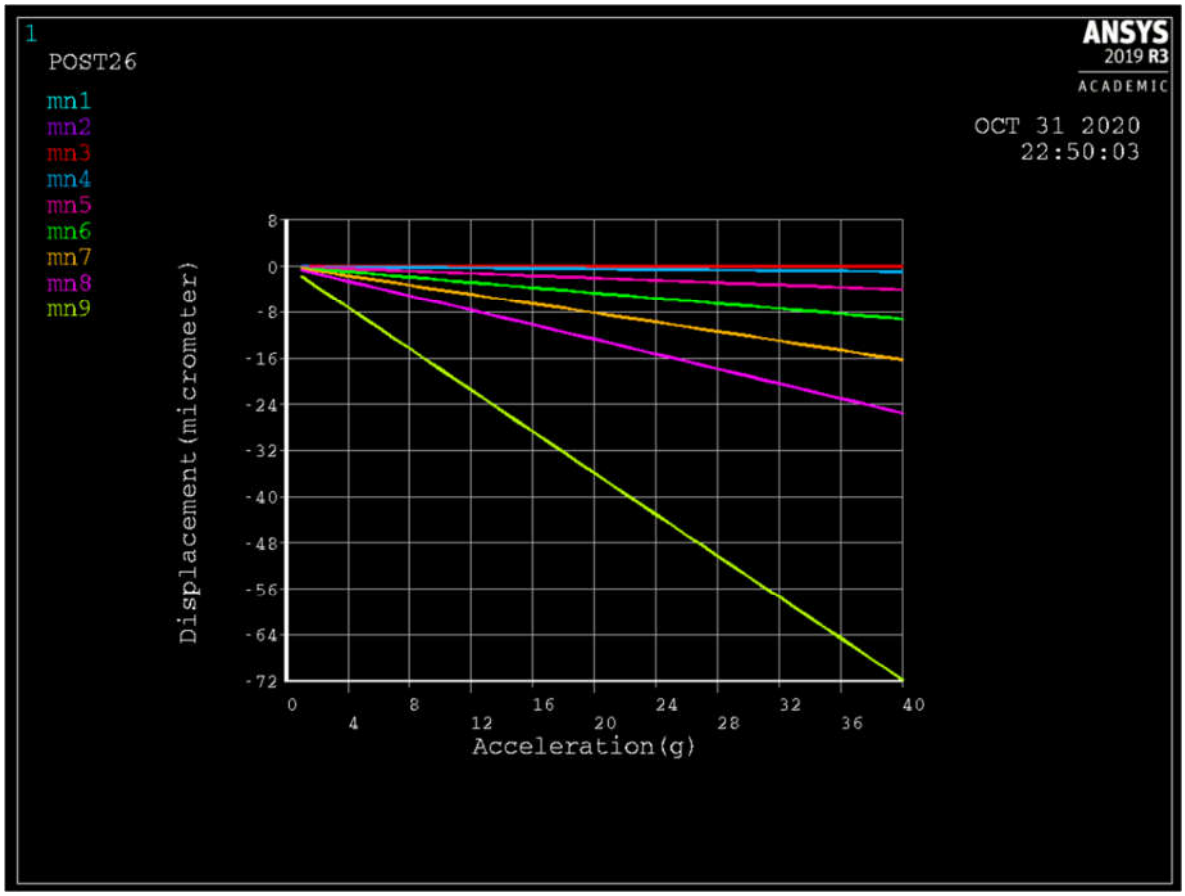


Figure 44: Master Node Displacements for acceleration ramp load

Similarly, a pressure ramp from 0 to 2.4 Pa was defined as a vector input to obtain the node displacements

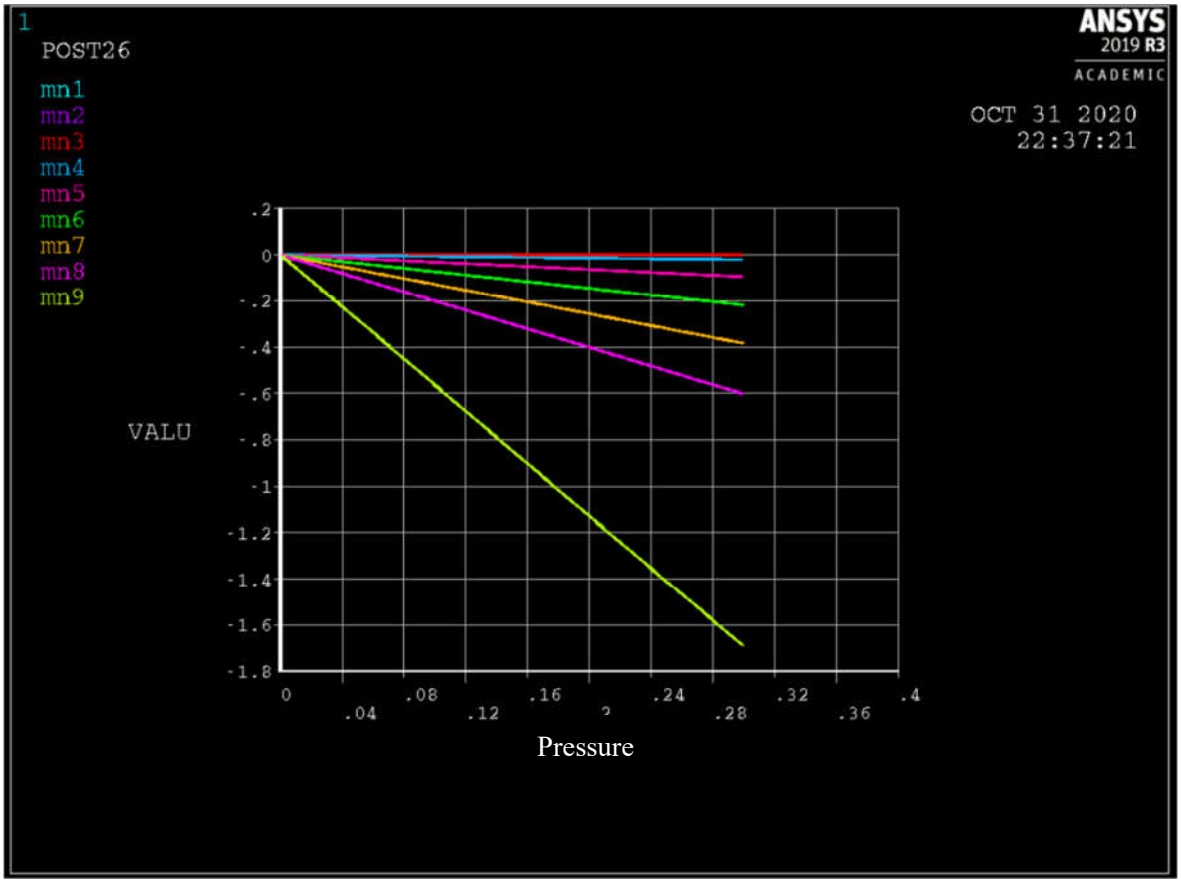


Figure 45: Master Node Displacements for acceleration ramp load

From the above figures it is seen that our main node of concern, mn8, positioned at the cantilever tip, provides sufficient displacement similar to theoretical calculations. Thus, values of displacement from mn8 can be used for subsequent calculations of piezoresistivity change in SSIT environment.

5.9 Determining desired excitation parameters for the MEMS device

Smart Systems Integrated Test Environment (SSIT) is an in-house developed software. In SSIT we aim to stimulate in behavioral level to perform system-level specifications. The ROM file generated from the ANSYS is uploaded to SSIT to perform ROM simulation and Analog simulation. The results of these simulations are used to develop the digital simulation of the system.

5.9.1 Simulation preparation- finding the equation for new resistance value

For nodal analysis, we chose node 28 as it is located on the junction of the cantilever and the suspended mass where the maximum strain and deflection of the beam is experienced. The maximum strain and deflection helps on determining the time of simulation. To start with SSIT examination on analog circuit, the change in resistance was calculated using the formula in equation (20). This was obtained from equations obtained in section 5.4.1.

$$\frac{\Delta R}{R} = \frac{3*U_x*b}{2*l^2} * E_{SiO_2} * \pi_t = (-195.195) * U_x \quad (20)$$

Where

B = cantilever thickness

L = length of the cantilever

E_{SiO_2} = Young modulus for SiO_2

$\pi_{\|}$ = piezo co-efficient for silicon

R = reference voltage

However for plotting the equation in SSIT, the R_{new} was calculated using the following equation

$$R_{new} = 30000 + (-195.195 * U_x) \quad (21)$$

5.9.2 MEMS ROM simulation analysis

The equation above was implemented in the SSIT source code for front impact on the sensor

```
SET,FIRST               !selecting the first solution subset
*DO,I,1,res
        *GET,uxval,NODE,28,UX   !getting the nodal displacement for the substep
        timestamp=t_res*I
        rval=30000-(-195.195*uxval)
        *VWRITE,timestamp,rval  !write to file
%G,%G
        SET,NEXT                !selecting the nex solution subset
*ENDDO
```

Figure 46: Code for SSIT

The following values were taken into consideration while performing the SSIT.

Table : MEMS ROM simulation settings

Parameter	Value
Simulation time [s]	300.00E-03
Impulse width [s]	100.00E-03
Resolution [s]	100.000E-06
Load [kPa]	2.12800E-03
Unused	1.000E+00

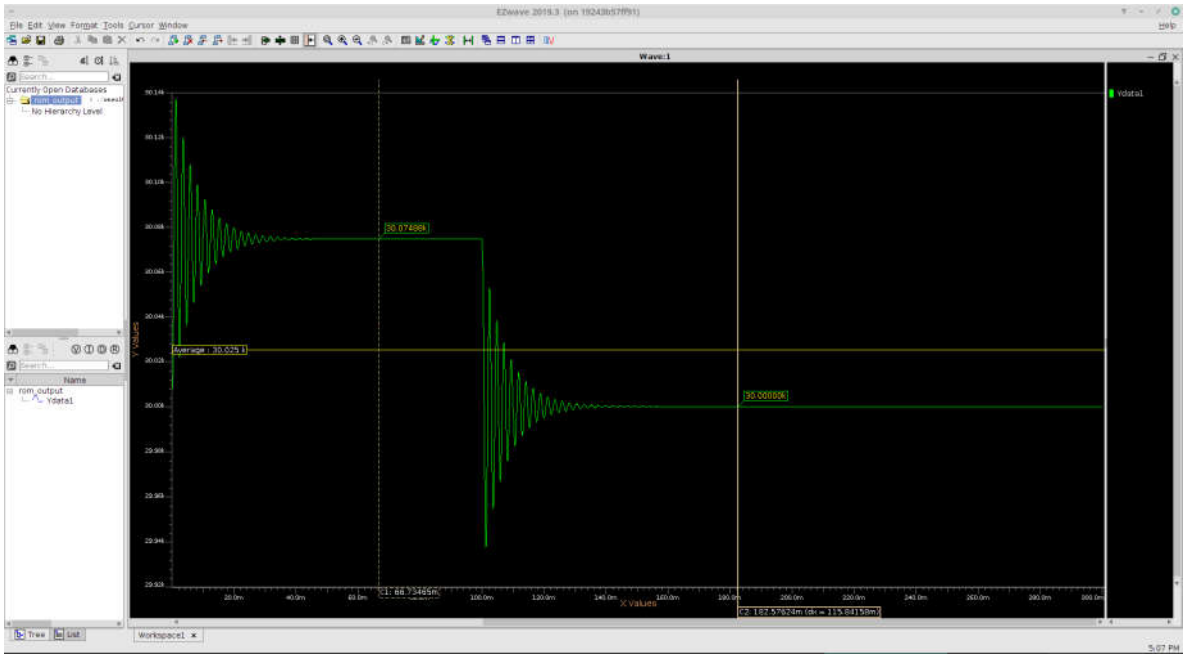


Figure47: Resistance change

In the above figure, it is possible to notice that the resistance is stable after 40ms which corresponds to resistance of 30.07488 KOhms. Therefore, the change in resistance (as calculated from the graph in Figure 47) is equal to:

$$\frac{\Delta R}{R} = \frac{30.07488 - 30.0000}{30.0000} = 0.249\%$$

which shows the resistance change in the range 0.2-0.4% fulfil the rules of a successful design.

5.10 Determining optimal parameters for the analog readout circuit.

The analog circuit consists of amplifier circuit and the VCO. The acceleration obtained from the accelerometer is mapped as the frequency change using a VCO. This frequency requires conditioning of the signal from the piezo resistor as ΔR. Using voltage divider rule this resistive drop is converted into a change in voltage. As this signal change is very less for a VCO to detect, the op-amp is used as an amplifier, remembering that the output of this VCO should always be in the range of the digital circuit.

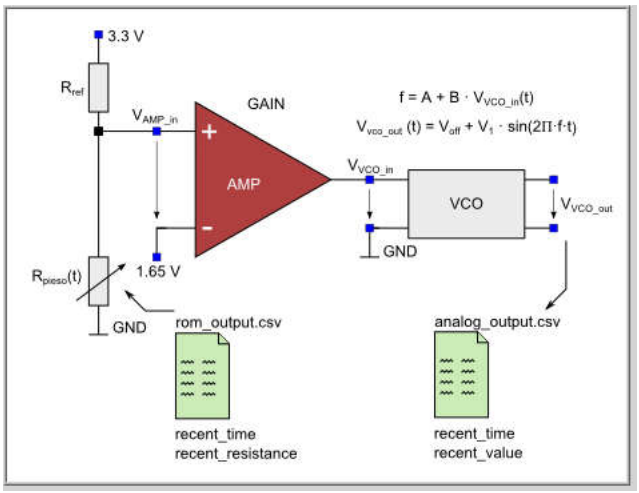


Figure 47: Analog circuit in SSIT

At 0 acceleration, the value of the piezo resistor is found to be 29KOhm. The amplifier input voltage is calculated using the equation given below. The voltage divider rule is applied to get that equation (22) where R_{ref} is equal to $R_{piezo.}$ The output of op-amp will be 0 when there is no deflection in the cantilever beam.

$$V_{amp_{in}} = \frac{Vdd * R_{piezo}}{R_{ref} + R_{piezo}} \quad (22)$$

$$V_{amp_{in}} = V + -V \quad (23)$$

The maximum change is resistance due to acceleration is 0.247% which is equal to 71ohms. This change in resistance changes the V_{amp_in} from 1.65V to 1.652036V.

The gain is calculated using the equation (24) given below to get an output voltage as -3V from the amplifier.

$$A = \frac{-V_{out}}{V^{+}-V^{-}} \quad (24)$$

$$=> \ -\frac{-3}{0.002036} = 1474$$

To determine the design parameters for VCO the following equation (25) is used

$$f = A + B * V_{vco_{in}} \quad (25)$$

Where A = tuning frequency,

B = tuning sensitivity,

V_{vco_in} = VCO's input voltage = amplifier's output.

The frequency range should be defined corresponding to V_{vco_in} = V_{amp_out} = -3 to 0 V. The digital system should be able to detect this frequency range. Considering this range, at the maximum frequency V_{vco_in} = 0V, therefore A = 10KHz, and at minimum frequency at V_{vco_in} = -3V so B = 5KHz.

So the VCO parameters are found to be:

A = tuning frequency = 10KHz.

B = tuning sensitivity = 5KHz.

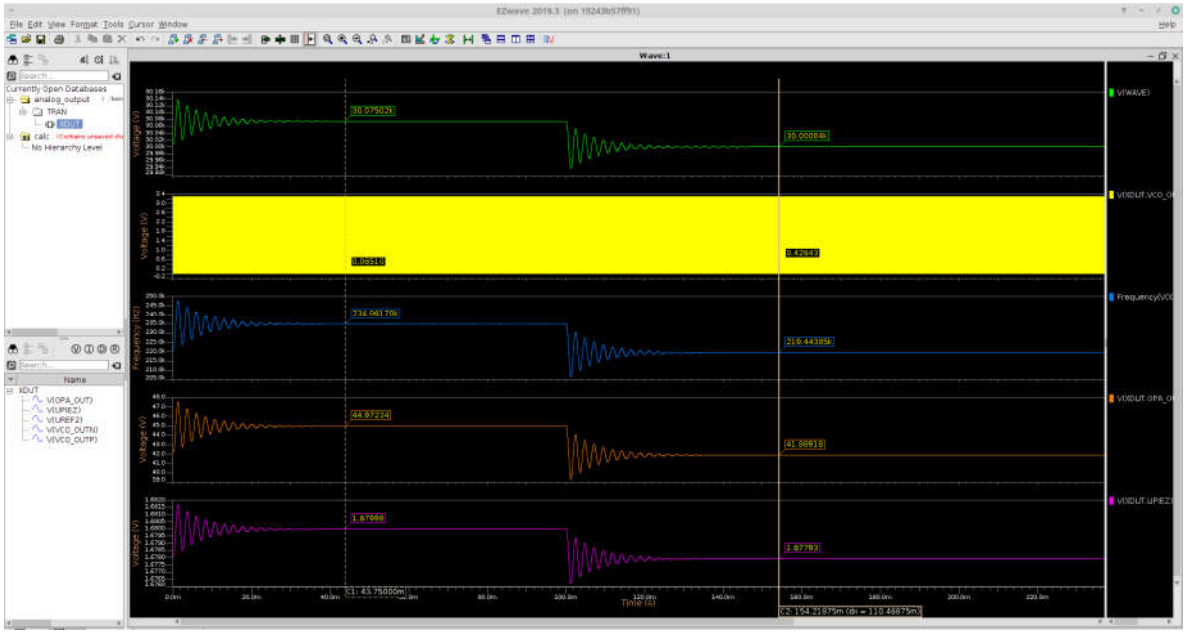


Figure 48: Analog simulations in SSIT

5.11 Determining parameters for digital circuit.

The digital circuit performs the fall detection and triggers alarm during a fall event. It receives filtered and amplified sensor signals from the VCO, and if the signal value and frequency surpass the derived thresholds for the project the digital circuit will send a signal to the alarm circuit to trigger the alarm. The alarm will run continuously for 10 minutes until a concerned person comes and stops the alarm using the stop button attached to the device. The time frame chosen earlier will be added with two sub-windows at its lower bound and upper bound, and that will be used to measure and analyze the acceleration data.

The VCO signal sampling rate will be determined using a counter. The measuring time window of the signal is crucial to the instant response of the device. The system frequency can be calculated using Nyquist sampling theorem:

$$f_s \geq 2 * F_{max}$$

using this formula, we choose our system frequency 2*247 Khz=494 Khz
However, corresponding to the clock generator for the designed hardware and to supply the system with safe frequency we are choosing 120 MHz as the system frequency.
Taking a time window of 0.5 ms and system frequency as 120 Mhz there will be around 60000 cycles per window. The input frequency from the analog circuit for max. acceleration will be 10 MHz, or 5000 clock cycles. A summary of the digital specification is given below in Table 8.

Table 8 Parameters for the digital circuit

Parameter	Value
System frequency	120 Mhz
Time window	0.5 ms
Input frequency range	10 MHz to 25 MHz
System cycles/window	6000
Cycles/window for Max acceleration	5000
Size of counter	14 bit

The algorithm for fall detection in the digital part is shown in the figure 29:

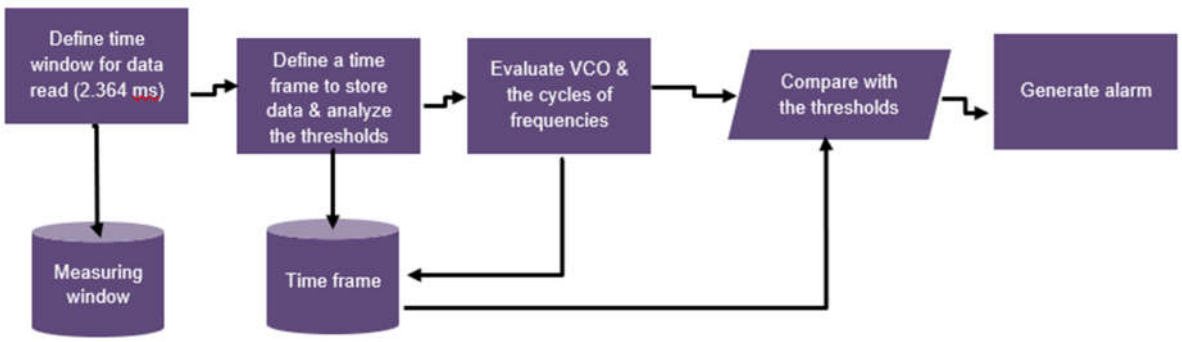


Figure 49: Algorithm for the digital circuit

5.12 Fabrication and Packaging of MEMS Device

5.12.1 Fabrication

Equipment required for the fabrication and packaging of the sensor include:

- Photolithography setup (Spinning unit, Heating unit for dehydration, UV light exposing Unit)
- Etching Tank for wet etching.
- STS Pegasus for DRIE or any other Bosch DRIE equipment.
- Measurement and characterization equipment for inspection of etch grooves and microstructure.
- CVD chamber for Deposition.

Materials required are silicon, aluminum for contacts and glass for packaging. Chemical Solutions used include acetone and distilled water for rinsing, 1-methoxy-2-propanol acetate (Developer solution), KOH for wet etching, acetone, Su-8 photoresist.

The following steps should be taken to fabricate the sensor: [26-33]

a. Wafer specification: The fabrication process starts with deciding the wafer specifications. We propose using a Double Sided Polished (DSP), n-type silicon wafer with 110 orientation. [33]
b. Oxidation (initial masking oxidation): This is growth of thin layer of SiO2 by means of Thermal Oxidation (wet and dry oxidation). For thick oxide layers, wet oxidation is preferred, but however we are required to grow a thin layer. We recommend that dry oxidation process be used, or quick dry oxidation done before and after wet oxidation.

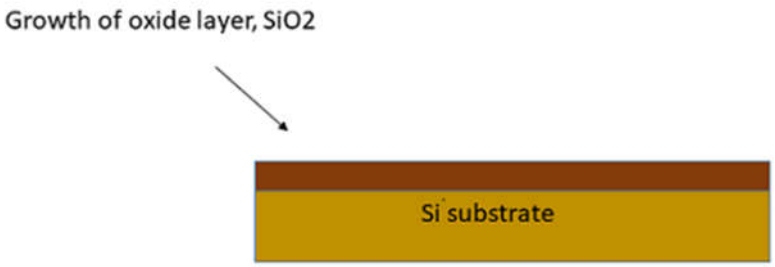


Figure 50: Oxidation

c. Next step is to create a Silicon on Insulation Layer by deposition of silicon. Deposition of Silicon is done using the Plasma Enhance CVD method in a vacuum and annealing afterwards.

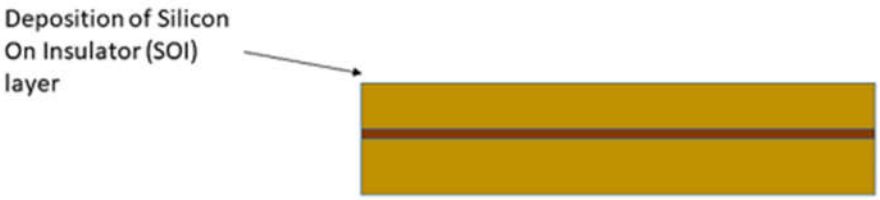


Figure 51: Deposition of Silicon on Insulator layer

d. The Piezoresistive layer is created by doping with Boron. This could be done by Ion Implantation.

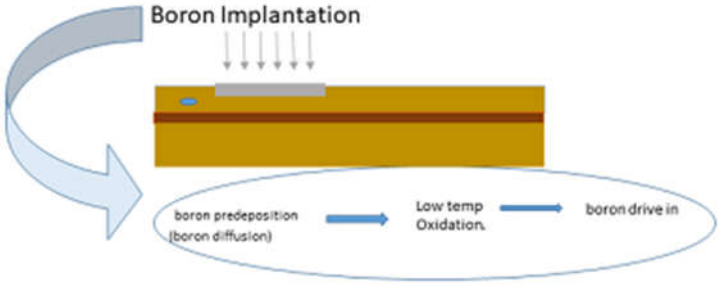


Figure 52: Ion implantation

e. The Aluminum pad for connection: Aluminum is used because of its good conductivity and ease of deposition. It has good bonding with SiO2. There are several methods for creating aluminum connection pads. We suggest the use of PVD methods such as sputtering or coating.

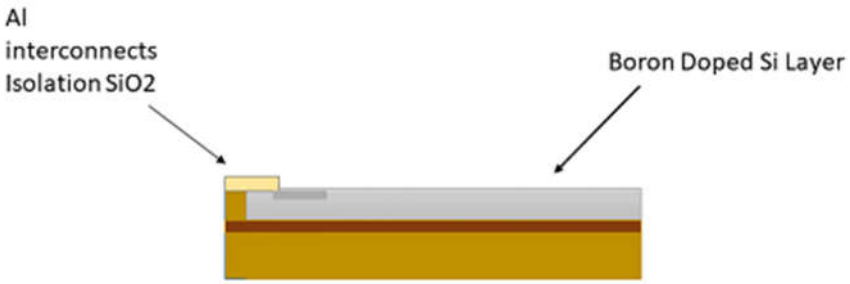


Figure 53: Aluminum pads for connection

f. Then the piezoresistive part is etched to specified thickness and dimensions according to our model.

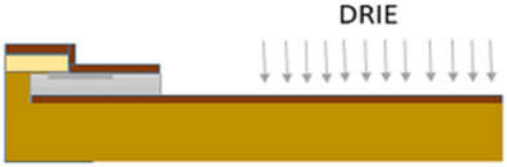


Figure 54: DRIE

g. The thin SiO2 layer now acts as protective layer for the aluminum pad and piezoresistive material.

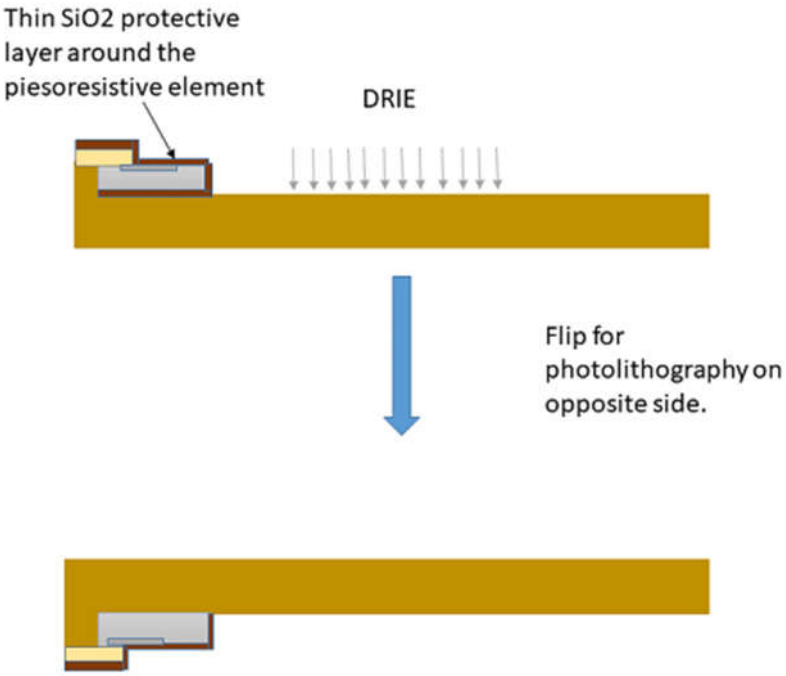


Figure 55: Protective layer for the aluminum pad and the piezoresistive material

h. The wafer is then flipped for wet etching on the opposite side. The mass dimension is used to create a chrome-on-glass mask which is used to conduct photolithography. The photolithography is done to be able to map out dimensions for etching and is shown in figures 57(a) and 37(b).

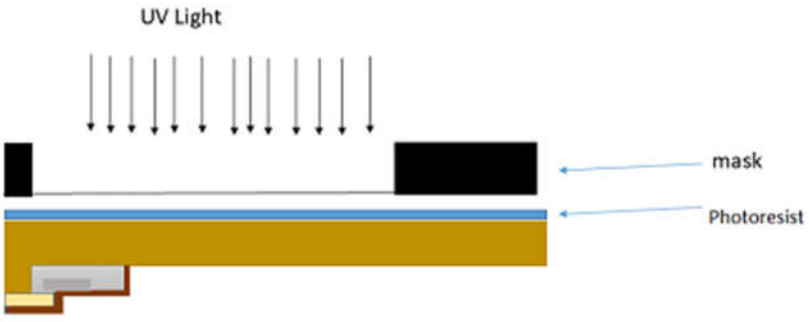


Figure 56(a): Photolithography on the opposite side using UV light

Figure 56(b): Wafer after photolithography

i. Wet etching using KOH is done to form the mass structure

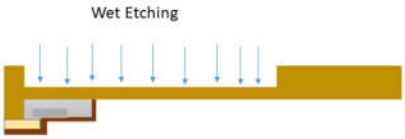


Figure 57(a): Wet etching using KOH

Figure 57(b): Etched wafer

i. The final step in fabrication is to observe the structure realized under a measurement and characterization equipment. We propose to get SEM images if possible to measure the actualized sensor dimension and compare against simulated dimension

5.12.2 Packaging:

For packaging, we suggest to use anodic bonding of glass to silicon. This gives a hermetic seal. We propose die/chip size packaging. [34]

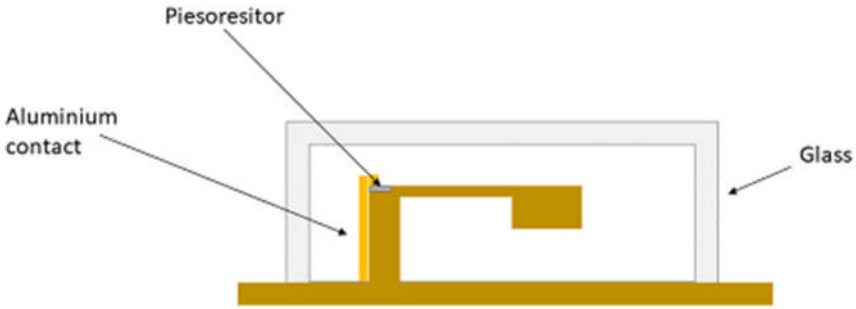


Figure 58: Packaged sensor

6 WP2: Analog circuit design

6.1 Overview

WP Leader	Pallavi	Rakin	Ankita	Faith
Ankita Rambhal	2	3	30	5
SUM Workload:	40	Duration:	14	

Objectives

- Designing an analog operational amplifier circuit which is able to gain the signal from the MEMS sensor and drive the A/D stage.
- Create an analog cell containing the application-specific amplifier circuit

Tasks

- Create the specification of the amplifier based on the output signal of the sensor and the input signal of the A/D converter (e.g.:amplification, bandwidth)
- Perform some hand calculations to determine the main properties of the amplifier
- Draw the schematic of the amplifier using Cadence Virtuoso
- Perform the required simulations
- Determine the main properties of the amplifier
- Perform the Monte-Carlo simulation to determine the input offset
- Create an analog cell for the amplifier
- Perform simulation using the amplifier in feedback configuration
- Create an analog cell for the amplifier with its feedback network
- Validate the operation of analog cells

Deliverables

- D2.1 The schematic of the amplifier with the size of the transistors and operating point parameters (model parameters, operating points of transistors)
- D2.2 Bode-plot of the amplifier (with and without frequency compensation)
- D2.3 The results of the transient analyses

6.2 Description of analog circuit design

6.3 Description of analog circuit design

After completion of MEMS design of the fall sensor, the next section deals with designing an analog circuit capable of translating the resistance change of the piezoresistor to voltage output. The piezoresistor is a part of a voltage divider that is connected to an Operational Amplifier. The change is piezoresistivity leads to a certain voltage output amplified by the Operational Amplifier. This is then fed to the input of a VCO whose instantaneous oscillation frequency is determined by the input voltage. This frequency enables the operation of the consequent digital circuit. The operational amplifier has been designed with LTSpice software here.

6.3.1 Absolute and relative mismatch

Major impediments to circuit design of converters, current mirrors, amplifiers, are component mismatch and noise. Even when components seem to be identically designed, such as fabricated and biased in the same conditions, they may not have same electrical properties. This can limit their precision and operation. [1] *"Mismatch is the process that can cause time dependent random variation in physical quantities of identically designed devices. Matching is the statistical study of the differences between identically designed components placed at a small distance in an identical environment and used with the same bias conditions." [2]*
There are two types of measuring a mismatch:

- Relative mismatch- Here, a difference between two parameters is given as a percentage.
- Absolute mismatch- the subtraction of two physical values.

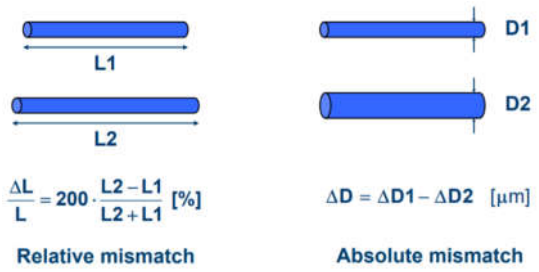


Figure: Relative and absolute mismatch [2]

Mismatches in identically designed components stem from stochastic or systematic effects. While stochastic effects lead to real mismatches, systematic effects result in offsets.

- Stochastic effects in matching
  These lead to random fluctuation of device properties and are termed true stochastic mismatches. This is inherent to a device and cannot be eliminated. The main causes can be irregularities in number of dopants in a channel, irregularities in oxide charges and interface states, polysilicon gate granularity, dimension effects and series resistances. [3]
- Systematic effects-
  These are differences that gain prominence at larger distances. The offsets stemming from such effects could be
  - Electrical- due to parasitics, for eg., line resistances.
  - Technological- these are produced during the fabrication process of wafers or etching.
  - Environmental- usually due to temperature effect on V_T and β parameters of Mosfets or due to mechanical stress effect from packaging [3]

Designing the circuit and circuit behaviour analysis with computer simulations can help to understand and get an estimation of such errors to an extent.

6.4 Specification of the analog amplifier

While designing the amplifier, the supply voltage was taken to be 3.3V as the nominal supply of the amplifier. The cut-off frequency should be higher than twice of the dominant mode frequency. The dominant mode frequency was 495Hz, hence the cut-off frequency was considered as 1KHz. As there is no negative feedback to the system the open loop gain was found to be 107 dB which is usually high. After providing the feedback, the amplification was found to be 102 dB. For the quick response, the system has the slew rate of 14.1 V/μs and the phase margin was encountered as 61.77° which ensures the stability of the system. Following table summarized the amplifier specifications.

Table: Specifications of the amplifier

Parameter	Value/Range
V_{DD}	3.3V
Open-loop gain	107 dB
Closed-loop gain	102 dB
Cut-off frequency	1KHz
Gain bandwidth	15MHz
Maximum input offset	0.38μV
Maximum input noise	100nV/sqrt(Hz)
Slew rate	14.1 v/μs
ICMR-	0.8

ICMR+	3
Phase margin	61.77°

6.5 Analog integrated amplifiers topologies

6.5.1 Overview

The Operational Amplifier, known as Op Amp, is a high gain DC differential amplifier. The Op Amp is very common in analogue circuitry because of its diverse functionality and its ability to be designed in different topologies and levels of complexities [1]. The perfect or ideal OpAmp, which is used/assumed in faster non-refined analysis of complex circuits involving OpAmps, is an operational amplifier with the following characteristics:

- Infinite open-loop gain
- Infinite input impedance
- Zero input offset voltage
- Zero output impedance
- Infinite bandwidth
- Zero power consumption

However, the ideal OpAmp is only ideal -as the name implies- and, therefore, rarely attainable. The real amplifier is more feasible and used in actual circuitry. The aim in designing an OpAmp is achieving said idealness; the ideal OpAmp is 'exemplary' and design of a real OpAmp would require lot of design trade-offs considerations (features to be considered at the expense of other features). A few of these trade-offs include: stability vs performance, power consumption vs noise and speed, and other conflicting parameters. [2, 3]
Designers hope to achieve an above-satisfactory power, output drive and noise performance by engaging relevant amplifier topologies for specified applications.

6.5.2 Operation of Selected Amplifier technologies

The following are the amplifier topologies that are associated with OpAmps and are used to achieve various functionalities in the analogue circuit.

i) Single Stage Topology

This is the simplest topology. Its simplicity offers high speed however lower functionality. The gain of a single stage OpAmp is low, and the mirror pole for a single-pole circuit is poor as well. A schematic representation of this topology is given below

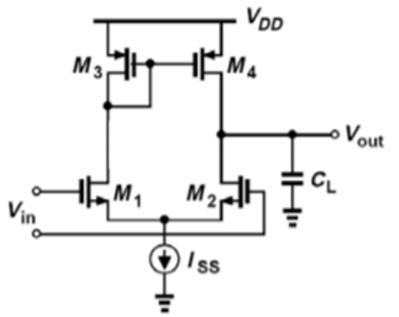


Fig x: Single Stage Amplifier

ii) Two Stage Topology

This topology aims to overcome the short coming of the single stage by introducing a second stage. This is the most popular topology used in OpAmp design. The two-stage just refers to the number of gain stages. This topology provides high gain and high output swing. A buffer at output only needed when resistive loads are used as driver rather than capacitive [4].

The classic Two stage Amplifier usually consists of a pmos differential pair with nmos current mirrors in first stage and common source amplifier for the second stage. [5]

An example of a two-stage amplifier identifying the stages is shown overleaf.

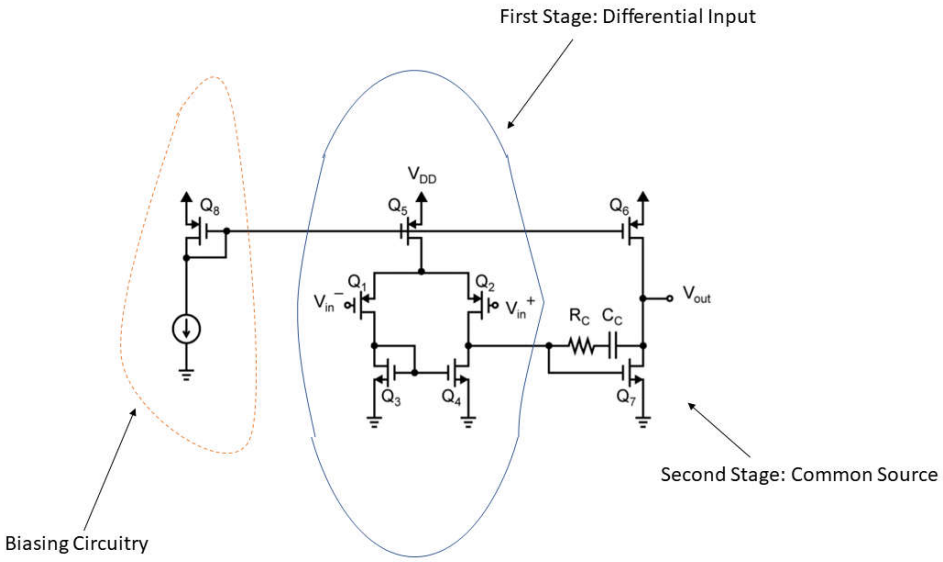


Fig. xi: Stages of a Classic Two-stage Amplifier

i) Telescopic Topology

This topology has a relatively high gain; it is important to note that there is difficulty in shorting the output to the input in this topology. Telescopic OpAmp offers the best compensation in the trade-off between gain, power dissipation and speed; however, the output swing is very limited.

The transformers are literally put on top of each other, hence the name telescopic. Although the speed is not as high as the single stage, there is less miller effect.

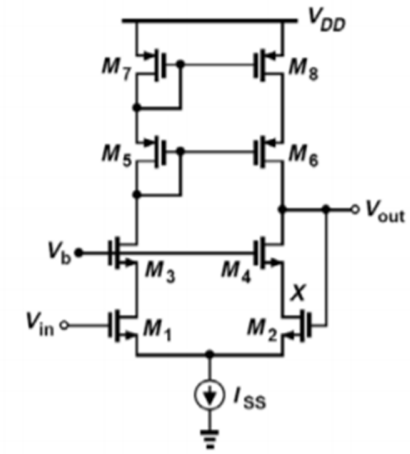


Fig xii: Telescopic cascaded Topology

ii) Gain Boosting Topology

This topology is majorly for increasing gain without decreasing output voltage swing. However, extra amplifiers used in achieving this might decrease the speed of the overall amplifier. An example of the gain boosting topology is shown in the schematic figure below.

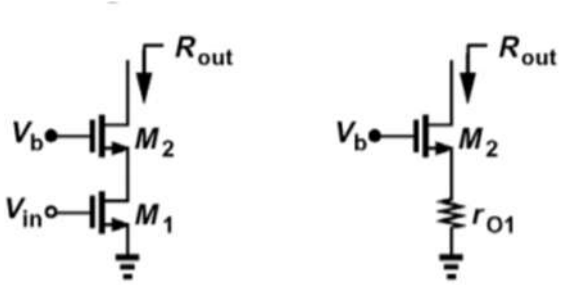


Fig xiii: Gain Boosting Topology

i) Folded Cascade

This works based on the property of a cascode to be folded. In comparison to the telescopic, it consumes more power and has greater noise contribution. This topology has fewer transistors at the output stage and a better frequency Power supply rejection Ratio.

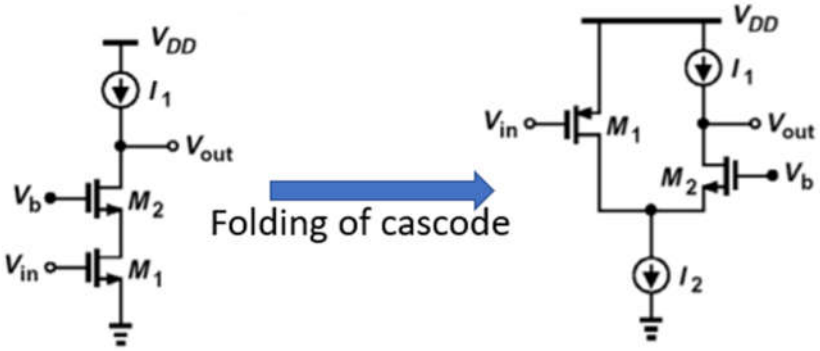


Fig xiv: Demonstration of folding of Cascode

Table 1: Summary of comparison between the topologies [1-6]

S/N	Amplifier Parameters	Amplifier Topologies				
		Single Stage	2-Stage	Gain-boosted	Telescopic	Folded cascode
	Gain	Low	High	High	Medium	Medium
	Output Swing	Medium	Highest	Medium	Medium	Medium
	Speed	High	Low	Medium	Highest	High
	Power Dissipation	Medium	Medium	High	Low	Medium
	Noise	High	Low	Medium	Low	Medium

6.5.3 Two-Stage Amplifier- Topology and Advantages

Considering the requirements of the fall detection system and above comparison table, the two - stage amplifier was chosen for this project. Two stage amplifiers was chosen because it has high gain, highest output swing and very low noise. Single stage amplifier was eliminated because it has very low gain which will not provide us the desired output for the system. Gain boosted amplifier also has the high gain but it has noise higher than the two stage amplifier. Among all the amplifier topologies, telescopic amplifier has the highest speed but medium gain. Hence two stage amplifier is the most suitable amplifier design for this project.

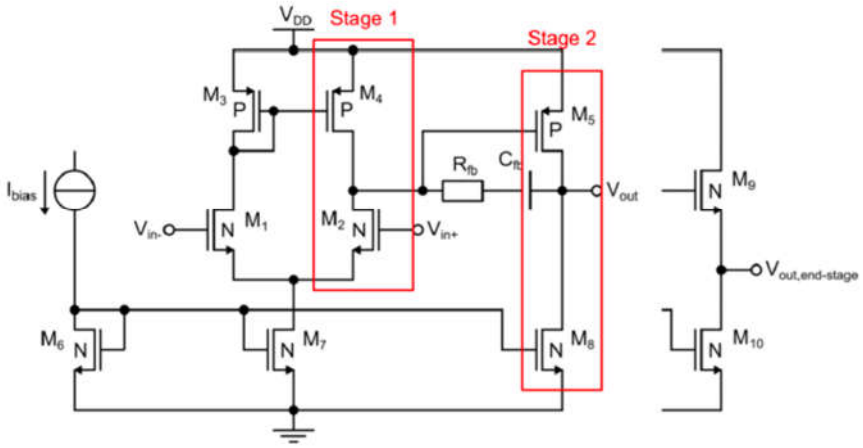


Figure xv : Two stage amplifier

Figure xv shows the structure of the two stage amplifier. The first stage provides a high gain whereas the second stage is usually designed to provide high swing. Transistors M9 and M10 are used as output buffer. Transistors M1 and M2 are differential pair of the first gain with current mirror formed by M3 and M4 as an active load. Transistor M7 provides the biasing to the entire amplifier. The second stage consists of common-source amplifier (M8) with active load M5. M5 is used for biasing the gate side.

6.5.4 Determining the transistor sizes and the biasing conditions.

For the design of the amplifier, we used Cadence Virtuoso with 350nm technology. Every technology has its own specific requirement. For this technology, the minimum channel length of the transistors must be higher than twice or equal to the 350nm. So, in the circuit, we have chosen Lmin as 1µm. dependent on the technologies, the parameters used for the calculations are given in the table below.

Table : Technology based transistor's paramteres

Parameter	**Value**
NMOS Gain factor (Kn)	170 µA/V^2
NMOS threshold voltage V_{tn}	0.5V

PMOS Gain factor (Kp)	60 μA/V^2
PMOS threshold voltage V_{tp}	0.75V

Using these values, the calculation was done as follows:

1. Tail Current Calculation

Considering F_{max} = 7.5KHz and load capacitance C_L = 10pF.
To achieve a good phase margin of 60°, C_C should be taken as ≥ 0.2 C_L.

$$C_C = 0.2 \times C_L = 0.2 \times 10p = 2pF$$

So,

$$I_{tail} = C_C \frac{dV}{dt} = C_C \times SR$$

But SR can be given as,

$$SR = 2\pi \times F_{max} \times V_{pp}$$

Here V_{pp} is taken as 3V, then SR = 14.1 V/μs.
Therefore, I_{tail} = 0.2μA.
Considering I_{tail} = 1.5μA just to make sure that SR will hold all the time.
Now, the transconductance can be given as

$$g_m 1 = \frac{I_{tail}}{V_{on}}$$

V_{on} is taken as 0.5V. Therefore $g_m 1 = 30\ \mu A/V$
And the gain bandwidth will be

$$GBW = \frac{g_{m1}}{C_C} = 15MHz$$

2. Size of the differential pair transistors

$$\left(\frac{W}{L}\right)_1 = \frac{\left(g_{m1}\right)^2}{2 \times I_{DS1} \times KPN} = 3.52 \approx 4$$

$$\left(\frac{W}{L}\right)_1 = \left(\frac{W}{L}\right)_2 = 4$$

3. Size of current mirror transistors

Since all the transistors should be in saturation region,

$$V_{GS1} \geq ICMR_+ - V_{t1}$$

ICMR+ was obtained as 3V after performing the dc simulation considering the size of the differential pair transistors calculated above.
Also, $V_{t1} = V_{tn} = 0.5V$
Therefore,

$$V_{GS1} \geq 3V - 0.5V = 2.5V$$

For M_3 transistor,

$$V_{GS3} = V_{dd} - V_{GS1} = 3.3V - 2.5V = 0.8V$$

So the current at transistor M_3 can be calculated as

$$I_3 = \frac{KPP}{2} \times \left(\frac{W}{L}\right)_3 \times (V_{GS3} - V_{t3})^2$$

But $I_3 = \frac{I_{tail}}{2} = 0.75\mu A$

So,

$$\left(\frac{W}{L}\right)_3 = 10$$

As the current passes through M_3, M_4, M_5 transistor is same, so their size must be equal.

$$\left(\frac{W}{L}\right)_3 = \left(\frac{W}{L}\right)_4 = \left(\frac{W}{L}\right)_5 = 10$$

For calculating current mirror $M_6 \; and \; M_7$ transistors V_{GS1} can be calculated from

$$I_{DS1} = \frac{KPN}{2} \times \left(\frac{W}{L}\right)_1 \times (V_{GS1} - V_{t1})^2$$

$$V_{GS1} = 0.54V$$

And $ICMR\_$ was obtained as 0.8V with the same analysis performed for ICMR+.

$$V_{DSat7} \leq 0.8V - 0.5V = 0.06V$$

Hence,

$$I_{DS7} = \frac{KPN}{2} \times \left(\frac{W}{L}\right)_7 \times (V_{GS7} - V_{t7})^2$$

$$\left(\frac{W}{L}\right)_7 = 4.24 \approx 5$$

$$\left(\frac{W}{L}\right)_6 = \left(\frac{W}{L}\right)_7 = 5$$

Since the current in second stage should be $I_{tail}/2$, the size of M_8 transistor should be the half of the size of M_7

$$\left(\frac{W}{L}\right)_8 = 2.5$$

The following table summarize the sizes of the transistors.

Table: Magnitude and aspect ratios of the transistors

No.	Type	Ratio	Length (µm)	Width (µm)
M1	NMOS	4	1	4
M2	NMOS	4	1	4
M3	PMOS	10	5	50
M4	PMOS	10	5	50
M5	PMOS	10	5	50
M6	NMOS	5	5	25
M7	NMOS	5	5	25

M8	NMOS	2.5	5	12.5
M9	NMOS	30	1	30
M10	NMOS	10	1	10

6.6 Schematic of the amplifier

Considering the above calculated values and the aspect ratios and bias of the transistor, the schematic was created as shown in figure {A]

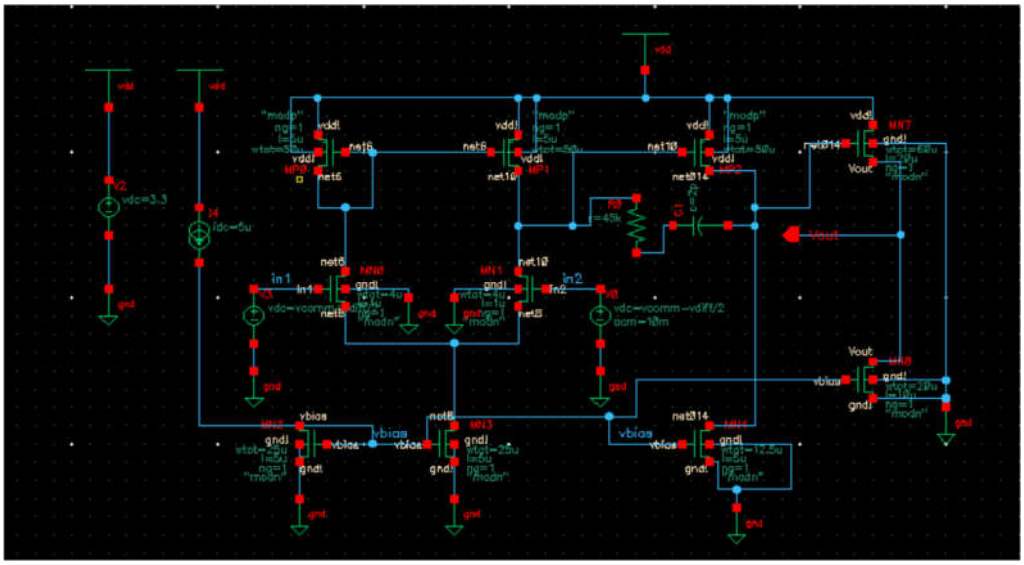


Figure A: Schematic of an amplifier

6.7 Performed analyses related to integrated amplifier

a. DC simulation and operating point

DC analysis was done to analyse the operating voltages of the transistors, mesh current ,branch current and to ensure ICMR range. It was also performed to verify that all the transistors are in saturation region.

While performing the DC sweep analysis, V_{DD} = 3.3V and I_{tail} = I_{bias} = 1.5µA was used. But the transistors were not in saturation region. To bring the transistors in saturation region, tail current was increased from 1.5µA to 5µA.

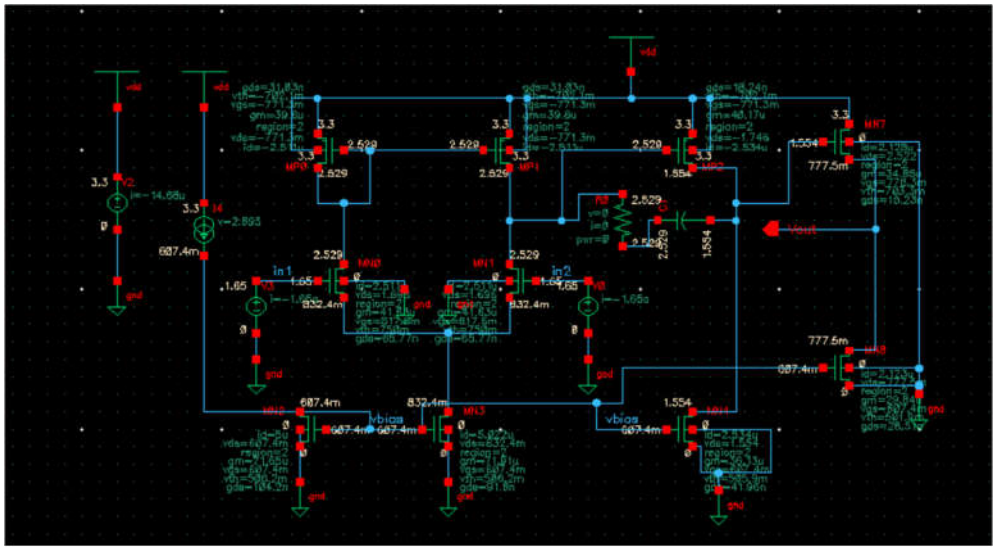


Figure B: Amplifier in Saturation region

The total gain of the system is calcultated using the formula

$$A_{tot} = A_1 + A_2$$

Where A1 is the gain of first stage and A2 is the gain of the second stage. These gains were calculated after performing the dc analysis at vdc = 1.65V.

Gain of first stage can be calculated as

g_{m1} = 41.63µ g_{ds2} = 65.77n

g_{ds4}= 31.03n

$$A_1 = \frac{g_{m1}}{g_{ds2} + g_{ds4}} = 430.06 = 52.67\ dB$$

Gain of second stage can be calculated as

g_{m2} = 36.13µ g_{ds8} = 41.96n

g_{ds5}= 18.24n

$$A_2 = \frac{g_{m2}}{g_{ds5} + g_{ds8}} = 598.5 = 55.54dB$$

Calculating total gain

$$A = A_1 + A_2 = 108\ dB$$

b. AC simulation

AC simualtion was performed to check the open loop gain of the amplifier and the phase margin. To perform the AC simulation, small ac magnitude of 10mV was given to the non-inverting terminal of the amplifier. After the analysis, the gain was obtained near 107 dB and phase margin was found to be 60°, which statisfies our requirement.. This shows that there is no need to tuning the amplifier.

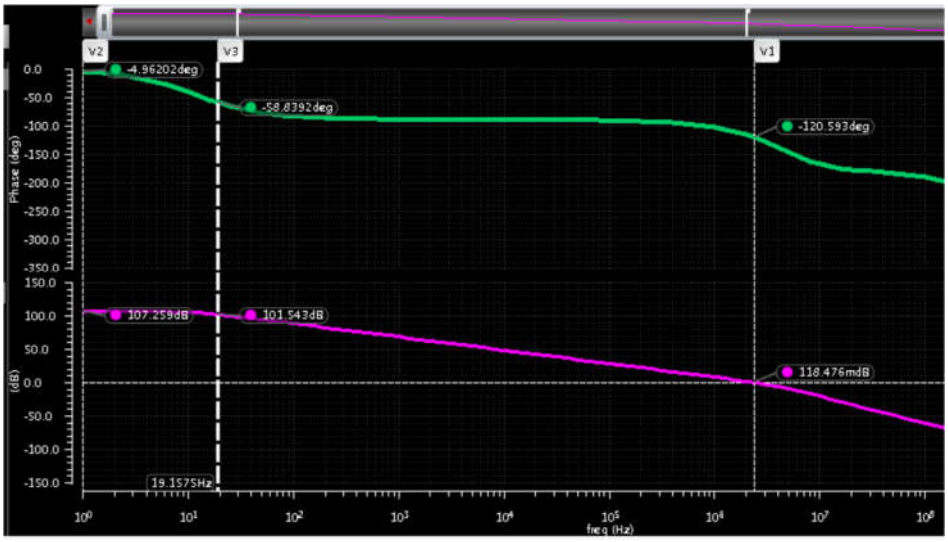


Figure C: Open-loop Phase and Gain plot

c. Transient simulation

Transient analysis was perform by converting the non-inverting terminal into sinusoidal voltage source with the amplitude of 10mV and 1KHX frequency.

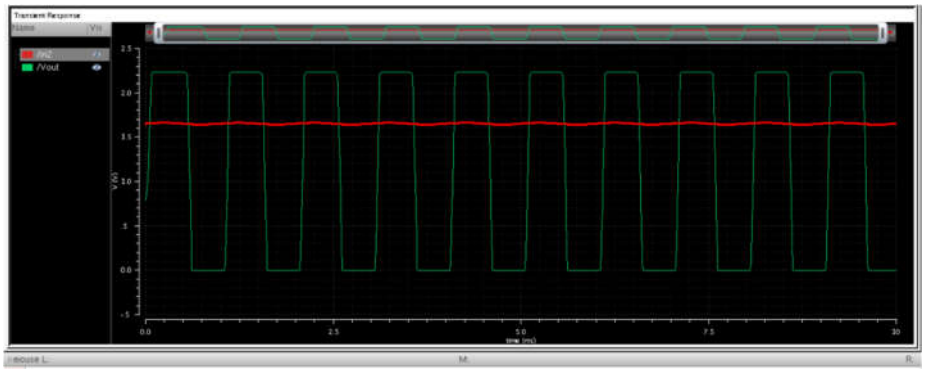


Figure D: Open loop Transient analysis

d. Stability simulation

For stability analysis, the amplifier was in closed loop from. A DC voltage source was added between Vout and inverting terminal of the amplifier as shown in the figure below.

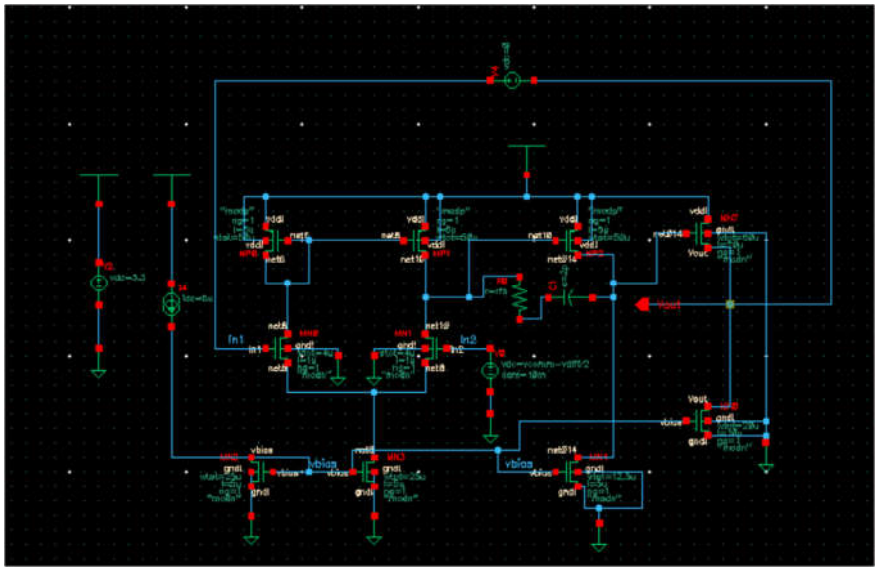


Figure E: Schematic for stability simulation

Stability analysis was done to obtain the closed-loop gain and phase margin of the system. The gain was found to be 102 dB and phase margin as 60°.

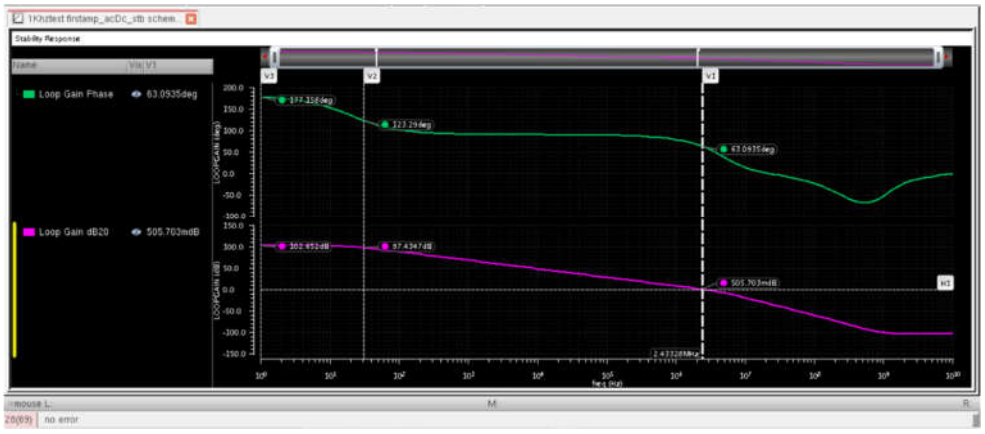


Figure F: Closed loop Phase and Gain plot

e. Monte-carlo simulation

Monte-carlo analysis was done to check the probability of different outcomes of the system. Following graph shows the outcome of the system at different conditions.

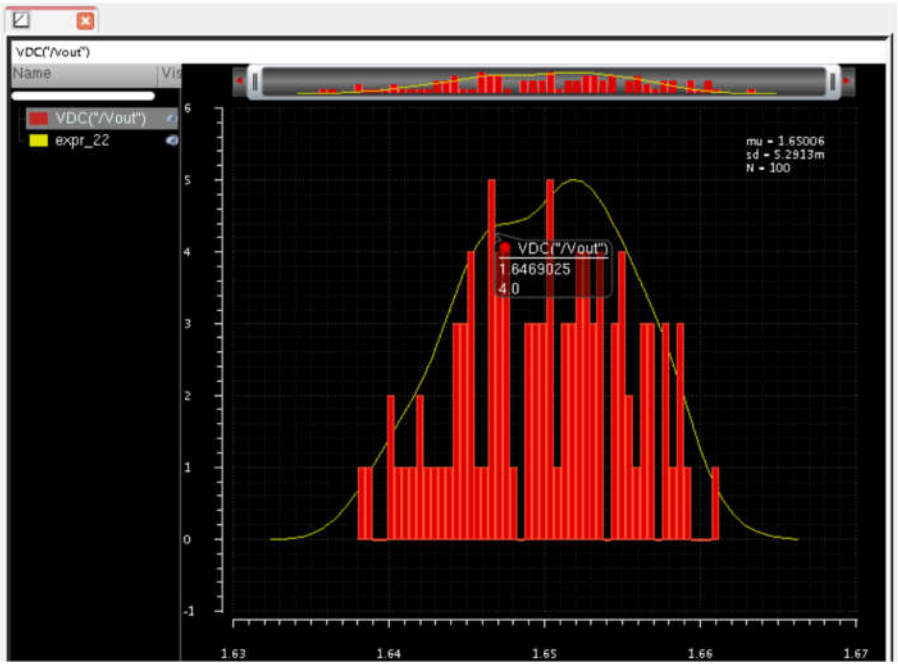


Figure G: Monte-Carlo simulation plot

f. Pvt

PVT simulation was done to test the system in best and worst condition and to verify whether the system will pass these conditions or not.

Test	Output	Nominal	Spec	Weight	Pass/Fail	Min	Max	C35B3_corner001	C35B3_corner00
1Khztest:firstamp_acDc_stb_pvt:1	Loop Gain	102.7 dB	> 80		pass	99.88 dB	105.2 dB	102.7 dB	103.9 dB
1Khztest:firstamp_acDc_stb_pvt:1	Phase Margin	62.24 degree	> 60		near	54.87 degree	66.79 degree	61.77 degree	66.27 degree

Figure H: PVT simulation results

From the results shown in the figures H and I, we can see that the amplification of the system can withstand the worst conditions, hence marked as **pass**. Whereas, the result for the phase margin test came as **near** which shows that the system just met the minimum condition. PVT simulation is also used to generate the datasheet of the system.

Results Summary ^Top

Test	Calculation	Expression	Target	Minimum Value	Maximum Value
1Khztest:firstamp_acDc_stb_pvt:1					
1Khztest:firstamp_acDc_stb_pvt:1	Loop Gain	ymax(db(mag(getData("loopGain" ?result "stb"))))	> 80	99.88 dB	105.2 dB
1Khztest:firstamp_acDc_stb_pvt:1	Phase Margin	getData("phaseMargin" ?result "stb-margin.stb")	> 60	54.87 degree	66.79 degree

Figure I: Result summary of PVT analysis

6.8 Analog cell creations and validation

After performing all the anaylses, analog cell was created. As shown in figure J, the amplifier was connect in negatice feedback with the input of 10mV. Resistors R0 and R1 are calculated using the formula

$$A_v = 1 + \frac{R1}{R0}$$

Here Av is the gain of the amplifier calculated in section 5.9 which was 1474.
Therefore R0 is taken as 5K and R1 as 295K.

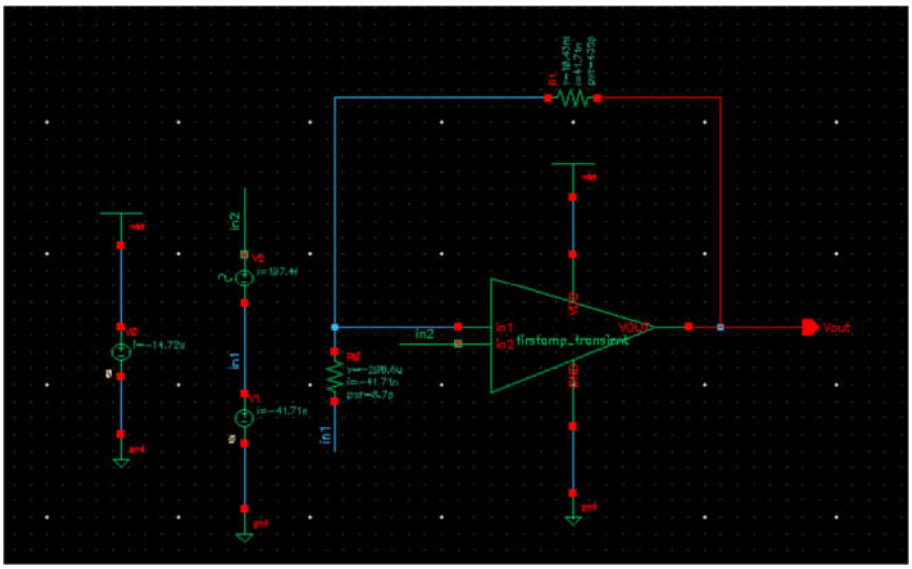


Figure J: Creation of analog cell

Figure I shows that the amplifier is working as per the requirements.

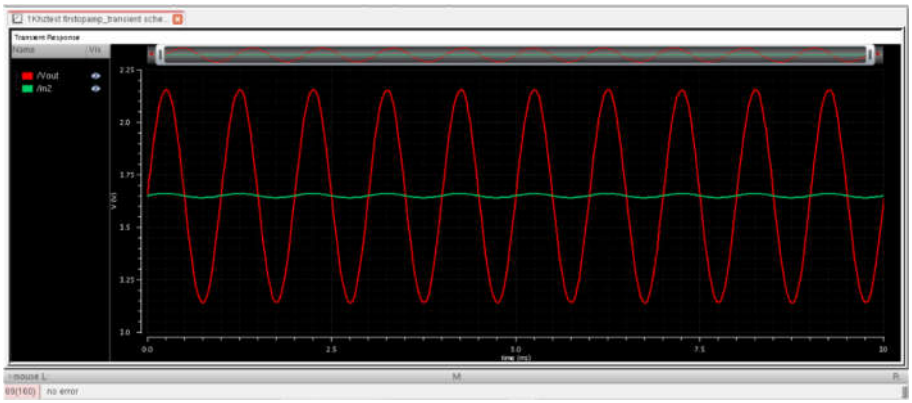


Figure K: Transient analysis

7 WP3: Digital circuit design

WP Leader	<member 1>	<member 2>	<member 3>	<member 4>
<member n>	10	10	10	10
SUM Workload:	50	Duration:	14	

Objectives

- Designing a digital circuit able to determine the frequency of the periodical signal provided by the A/D converter and implementing a decision circuitry able to generate an interrupt for the host microprocessor based on the acquired frequency measurement data.

Tasks

- Create a RTL model of the circuit using the synthesizable subset of VHDL.
- Perform functional verification using ModelSim-Altera with a VHDL-based test environment.
- Synthesize the circuit onto FPGA technology using Altera Quartus II.
- Perform static timing analysis (STA) using Quartus II.
- Generate the post-place&route simulation model of the circuit using Quartus II. Perform timing simulation using ModelSim-Altera with the VHDL-based test environment used in the functional verification phase.

Deliverables

- D3.1 The RTL model of the circuit and the test environment (VHDL source code).
- D3.2 Functional verification results.
- D3.3 Synthesis summary (resource usage).
- D3.4 Post-place&route (timing) simulation results.

8 7.1. Determining the system frequency

In the analog design, we designed the analog cell that contained the op-amp circuit to derive the gain and perform the A/D operation of our designed MEMS sensor. However, we designed the circuitry for one sensor only, while in our case we would be requiring tri-axial acceleration measurement. Designing based on one sensor only was adequate in the previous stages, because the tri-axial acceleration measurement approach will be three-stage scaled up version of an analog cell containing one sensor, because the threshold and design of only one cell will be similar to the other cells. [FIGURE] explains our proposed measurement approach. As you can see from [FIGURE b], the single-axis sensors are assumed to be placed in parallel to the X-Y-Z planes of the system, in order to combinedly measure the acceleration along all the directions.

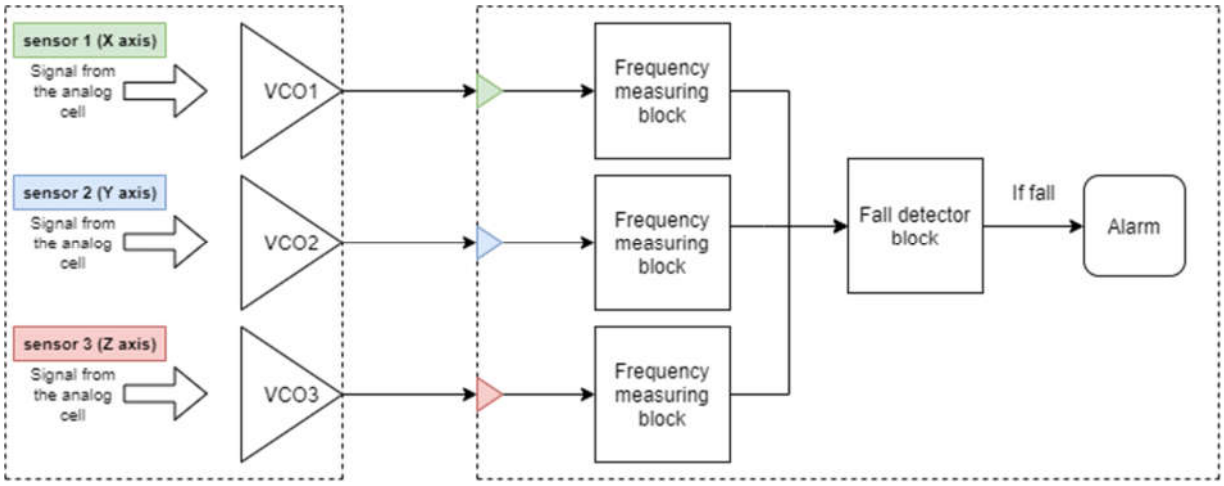


Figure_: Overview of Sensor Design

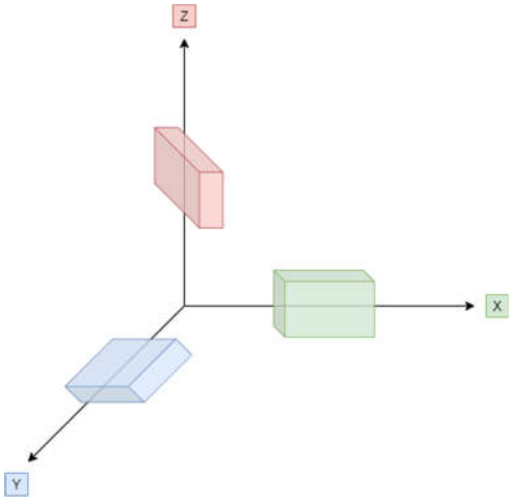


Figure_: Orientation of three sensors for fall detection

However, the major implementation will be required in the digital system design part, where we must consider tri-axial acceleration data. Because our proposed algorithm that detects fall and ADL (Activities of Daily Living) depends upon this parameter. This means that the digital system will receive three frequency inputs from three VCOs of the analog cells. [FIGURE]

The designed digital circuit (which will be discussed in this section) is able to detect an ADL event and can also identify a fall event. On a holistic level, the proposed system operates as follows:

- Count the number of high edges of each of the three signals within the defined time measurement window
- If the RMS value of the number of edges is higher than the lower-bound threshold but lower than the higher-bound threshold then it is an ADL event.
- If the RMS value of the number of edges is higher than the higher-bound threshold then it is a fall event, and an output signal is passed to the alarm (buzzer).

Based on our derivation of the optimal parameters for the digital system in section 5.9 and 5.10, the digital system has been designed to detect frequencies in the range from 10MHz to 25 MHz. Thus, the bandwidth of the system is 15 MHz. Following the discussion regarding the Nyquist-Shannon's sampling theorem, the minimum choice of system frequency should be 50 MHz, that means that our system should be able to operate properly with minimum frequency of 50 MHz. However, a system frequency of 120MHz was set to provide the system to use the clock generator in a safe margin, corresponding to an easily available standard on the market. This means that with our defined time window of 0.5 ms there will be 60000 cycles (rising edges) in total. The following edge counter thresholds have been considered:

- If the rms_edge counter is equal to 5000 (corresponding to 10 Mhz) for than one-time window, then the system will initially characterize it as an ADL.
- If the rms_edge counter is equal to 5000 (corresponding to 10 Mhz) for 5-time windows, then the system will characterize it as a fall event and the alarm will trigger.

Value 5000 corresponds to 10 MHz of analog design, the highest acceleration value to be considered for a fall, and the corresponding RMS of the three counters crossing this threshold.

9 7.2. Limitation for fall detection using Cyclone III and Quartus II

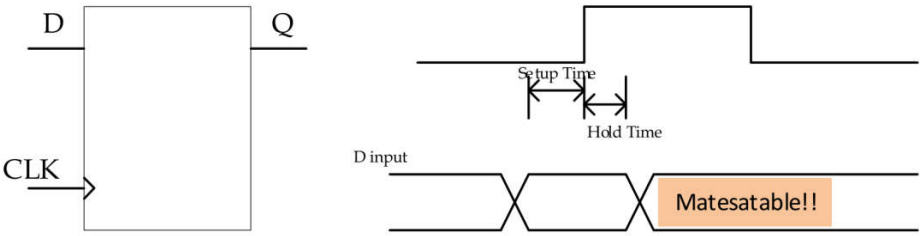
Following the discussion of the fall detection algorithm presented in section [SECTION], we proposed three characteristics to be measured to detect falls, they are: RMS, pitch angle and roll angle. Although we can measure and synthesize RMS value, we could not design a synthesizable pitch angle and roll angle measurement mechanism. This is because "real" number or "real" mathematical operations are not synthesizable. Also, fixed-point function definition using ALTERA CORDIC_IP to get arctan value based on the CORDIC algorithm is not supported for Cyclone III (EP3C16F484) via Quartus II.

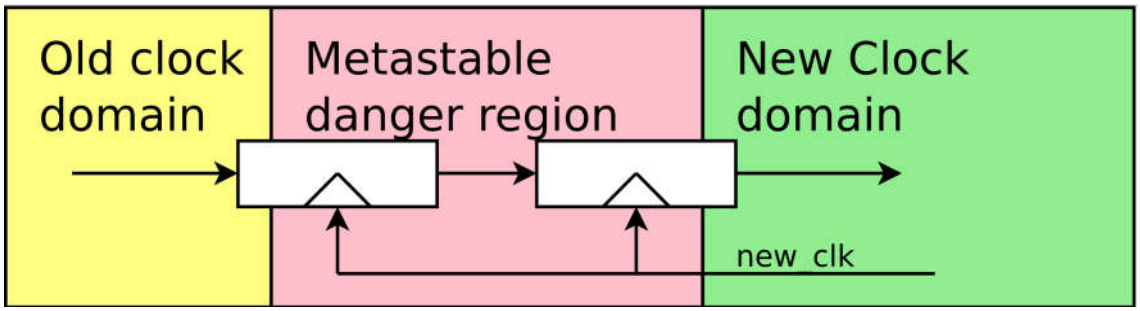
107.3. RTL model of frequency measurement circuit

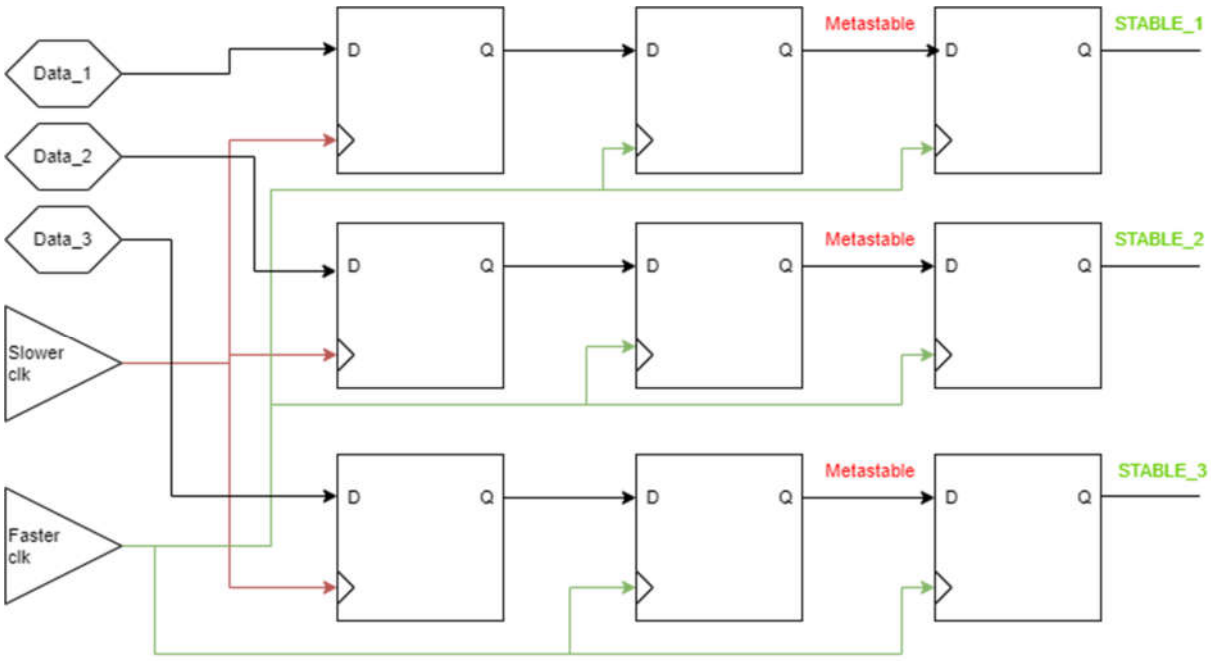
10.1 7.3.1. Frequency measurement implementation

The 3-channel frequency measurement circuit which characterizes between an ADL event and a fall event is discussed in this section. The frequency measurement approach chosen for the project was based on a double flip-flop synchronization mechanism [PAPER]. The mechanism is described in figure [FIGURE] This mechanism states that, a minimum of 2 stages of re-synchronization flip-flops are included in the destination domain to avoid issues with metastability in the destination domain [PAPER]. Different clock domains have clocks which have a different frequency, a different phase (due to either differing clock latency or a different clock source), or both. Either way the relationship between the clock edges in the two domains cannot be relied upon. This is beneficial over a single stage flopping, where two asynchronous resets could have been considered via an OR gate. Although the single-stage process is simpler, it will eventually cause:

- Timing anomaly in the measurement
- Clock domain crossing (Metastability) [FIGURE]







117.4. State machines for frequency measurement

To implement the double-flopping synchronization during edge counting and timer measurement, two procedures have been followed. They are:

- State machine
- Hand-shaking protocols between the df blocks

State machine divides the edge counting into two logical stages, and hand-shaking protocol establishes the communication path between the states. The states are:

State_edge_counter	**Description**
Wait_for_start_measure	Waiting for counting flag to be triggered
Counting	Main edge counting stage, checks ack bit at the same time. If ack is not received, then the edge counter returns to 'wait_for_start_measure' stage

The resource declaration for the edge counter FSM is given in figure [FIGURE]. As we can see, resources have been assigned for all the three-frequency measurement sub-systems.

```
-- edge counter resources ---------------------------------------------------------
TYPE state_edge_counter_type IS (wait_for_start_measure, counting);

---------------------
----1st frequency----
---------------------
SIGNAL state_edge_counter : state_edge_counter_type := wait_for_start_measure;
SIGNAL edge_counter : INTEGER RANGE 0 TO edge_counter_max := 0;

---------------------
----2nd frequency----
---------------------
SIGNAL state_edge_counter_2 : state_edge_counter_type := wait_for_start_measure;
SIGNAL edge_counter_2 : INTEGER RANGE 0 TO edge_counter_max := 0;

---------------------
----3rd frequency----
---------------------
SIGNAL state_edge_counter_3 : state_edge_counter_type := wait_for_start_measure;
SIGNAL edge_counter_3 : INTEGER RANGE 0 TO edge_counter_max := 0;
```

The timer measurement is also divided into four stages. The states are:

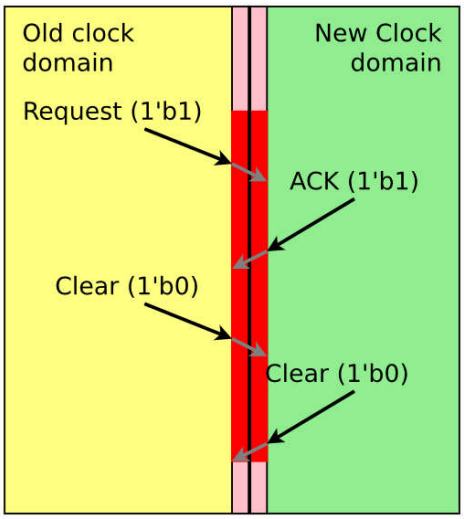
State_measure_timer	Description
Wait_for_enable_measure	Wait until enable measure flag is triggered
Wait_for_ack_1	Checks if the 2nd ack for edge counter is received. If true, then move on to measure_timing, else wait
Measure_timing	Increase time counter if time_counter is less than time_counter_max. If time_counter is equal to time_counter_max, then reset time counter and enable_edge_counter flag, and move to wait_for_ack_2 state
Wait_for_ack_2	If the 2nd ack for edge counter received is 1, then wait until it turns to zero. When it becomes zero, then trigger transmission flag, assign new_data if fall_adl_detected is 00 and return to the initial state of the timer

The resource declaration for the timer measurement FSM is given in figure [FIGURE]. Like the edge counting section, we can see resources has been assigned for all the three-frequency measurement sub-systems.

```
-- measure timer resources ----------------------------------------------------------------
TYPE state_measure_timer_type IS (wait_for_enable_measure, wait_for_ack_1, measure_timing, wait_for_ack_2);
--1st frequency timing measuring FSM
SIGNAL state_measure_timer : state_measure_timer_type := wait_for_enable_measure;
--2nd frequency timing measuring FSM
SIGNAL state_measure_timer_2 : state_measure_timer_type := wait_for_enable_measure;
--3rd frequency timing measuring FSM
SIGNAL state_measure_timer_3 : state_measure_timer_type := wait_for_enable_measure;

--time counters-------------------------------------
----1st frequency----
SIGNAL time_counter : INTEGER RANGE 0 TO time_counter_max := 0;
----2nd frequency----
SIGNAL time_counter_2 : INTEGER RANGE 0 TO time_counter_max := 0;
----3rd frequency----
SIGNAL time_counter_3 : INTEGER RANGE 0 TO time_counter_max := 0;

--1st frequency data and sending flag
SIGNAL send_trigger : STD_LOGIC := '0';
SIGNAL new_data : unsigned (15 DOWNTO 0) := (OTHERS => '0');

--2nd frequency data and sending flag
SIGNAL send_trigger_2 : STD_LOGIC := '0';
SIGNAL new_data_2 : unsigned (15 DOWNTO 0) := (OTHERS => '0');

--3rd frequency data and sending flag
SIGNAL send_trigger_3 : STD_LOGIC := '0';
SIGNAL new_data_3 : unsigned (15 DOWNTO 0) := (OTHERS => '0');
```

127.5. Handshake protocol between the state machines

[FIGURE] presents the working principle of the hand-shake protocol for a single frequency measurement. As discussed above, two clock domains have been fitted (yellow: the old clock, green: the new clock) with a metastable region (the pink-red region).



```
-- edge counter <-> measure timer ------------------------------------------------------------
-- Handshake protocol resources --------------------------------------------------------------
--1st frequency--
SIGNAL enable_edge_counter : STD_LOGIC := '0';
SIGNAL enable_edge_counter_ack : STD_LOGIC := '0';

SIGNAL df_enable_edge_counter_1 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_2 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_1 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_2 : STD_LOGIC := '0';

--2nd frequency--
SIGNAL enable_edge_counter_2 : STD_LOGIC := '0';
SIGNAL enable_edge_counter_ack_2 : STD_LOGIC := '0';

SIGNAL df_enable_edge_counter_1_2 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_2_2 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_1_2 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_2_2 : STD_LOGIC := '0';

--3rd frequency--
SIGNAL enable_edge_counter_3 : STD_LOGIC := '0';
SIGNAL enable_edge_counter_ack_3 : STD_LOGIC := '0';

SIGNAL df_enable_edge_counter_1_3 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_2_3 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_1_3 : STD_LOGIC := '0';
SIGNAL df_enable_edge_counter_ack_2_3 : STD_LOGIC := '0';
```

The steps of the protocol are as follows:
Step-1: Initial request for data transfer is sent from the old clock to the new clock domain passing the metastability region.
Step-2: The new clock receives the request and sends an ack signal back.
Step-3: The old clock receives the ack from previous step and finally clears the request signal
Step-4: The new clock sends an acknowledgement back when the data is cleared through its block.

```
-- Functions for the implementation of the handshake protocol for 3rd frequency
-- Step-1: Initial request for data transfer is sent from the old clock to the new clock domain passing the metastability region.
-- Step-2: The new clock receives the request and sends an ack signal back.
-- Step-3: The old clock receives the ack from previous step and finally clears the request signal
-- Step-4: The new clock sends an acknowledgement back when the data is cleared through its block.
L_DF_ENABLE_EDGE_COUNTER_ACK_3 : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_ack_1_3 <= '0';
        df_enable_edge_counter_ack_2_3 <= '0';
    ELSIF (rising_edge(clk)) THEN
        df_enable_edge_counter_ack_1_3 <= enable_edge_counter_ack_3;
        df_enable_edge_counter_ack_2_3 <= df_enable_edge_counter_ack_1_3;
    END IF;
END PROCESS;

L_DF_ENABLE_EDGE_COUNTER_3 : PROCESS (freq_in_3, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_1_3 <= '0';
        df_enable_edge_counter_2_3 <= '0';
    ELSIF (rising_edge(freq_in_3)) THEN
        df_enable_edge_counter_1_3 <= enable_edge_counter_3;
        df_enable_edge_counter_2_3 <= df_enable_edge_counter_1_3;
    END IF;
END PROCESS;

-- Functions for the implementation of the handshake protocol for 2nd frequency
-- Step-1: Initial request for data transfer is sent from the old clock to the new clock domain passing the metastability region.
-- Step-2: The new clock receives the request and sends an ack signal back.
-- Step-3: The old clock receives the ack from previous step and finally clears the request signal
-- Step-4: The new clock sends an acknowledgement back when the data is cleared through its block.
L_DF_ENABLE_EDGE_COUNTER_ACK_2 : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_ack_1_2 <= '0';
        df_enable_edge_counter_ack_2_2 <= '0';
    ELSIF (rising_edge(clk)) THEN
        df_enable_edge_counter_ack_1_2 <= enable_edge_counter_ack_2;
        df_enable_edge_counter_ack_2_2 <= df_enable_edge_counter_ack_1_2;
    END IF;
END PROCESS;

L_DF_ENABLE_EDGE_COUNTER_2 : PROCESS (freq_in_2, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_1_2 <= '0';
        df_enable_edge_counter_2_2 <= '0';
    ELSIF (rising_edge(freq_in_2)) THEN
        df_enable_edge_counter_1_2 <= enable_edge_counter_2;
        df_enable_edge_counter_2_2 <= df_enable_edge_counter_1_2;
    END IF;
END PROCESS;
```

```
-- Functions for the implementation of the handshake protocol for 3rd frequency
-- Step-1: Initial request for data transfer is sent from the old clock to the new clock domain passing the metastability region.
-- Step-2: The new clock receives the request and sends an ack signal back.
-- Step-3: The old clock receives the ack from previous step and finally clears the request signal
-- Step-4: The new clock sends an acknowledgement back when the data is cleared through its block.
L_DF_ENABLE_EDGE_COUNTER_ACK_3 : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_ack_1_3 <= '0';
        df_enable_edge_counter_ack_2_3 <= '0';
    ELSIF (rising_edge(clk)) THEN
        df_enable_edge_counter_ack_1_3 <= enable_edge_counter_ack_3;
        df_enable_edge_counter_ack_2_3 <= df_enable_edge_counter_ack_1_3;
    END IF;
END PROCESS;

L_DF_ENABLE_EDGE_COUNTER_3 : PROCESS (freq_in_3, reset)
BEGIN
    IF (reset = '1') THEN
        df_enable_edge_counter_1_3 <= '0';
        df_enable_edge_counter_2_3 <= '0';
    ELSIF (rising_edge(freq_in_3)) THEN
        df_enable_edge_counter_1_3 <= enable_edge_counter_3;
        df_enable_edge_counter_2_3 <= df_enable_edge_counter_1_3;
    END IF;
END PROCESS;
```

The major advantage of implementing this method is that the system will only start reading when it is ready to receive, that means has the control over receiving data by not allowing to post new requests unless the acknowledgement stages are cleared. Thus, the system will always receive stable data for measurement. The following code defines the two stages of double-flopping of the hand-shake signals for all the three subsystems of frequency measurements [FIGURE] to [FIGURE]. As we can, proper synchronization and stability between the edge counter and time measuring state machines can be implemented.

137.6. Edge Counting Process

The codes for edge counting for the three frequency subsystems are given below in [FIGURE]. As mentioned earlier, the state machine for edge counting consists of two states: wait_for_start_measure and counting. The first state defines the wait case of a subsystem to start the frequency measurement, while the second state defines the state of counting. Each of the frequencies are measured based on the amount of rising edges in the specified time window:

$$signal_{frequency} = \frac{no\ of\ rising\ edges}{time\ window}$$

The counting operation for the edge counter only works when it is enabled via the time state machine via the hand-shaking protocol. As we can see in the code, counter enable ack is activated, and after receiving a return acknowledgement to clear the request, the counting process starts and continues until the counter reaches the maximum threshold value 12500.

```
186  --1st frequency input--
187  -----------------------
188  -- FSM for edge counter for 1st frequency
189  -- Wait_for_start_measure: Waiting for counting flag to be triggered
190  -- Counting: Main edge counting stage, checks ack bit at the same time.
191  -- If ack is not received, then the edge counter returns to 'wait_for_start_measure' stage
192
193      L_EDGE_COUNTER : PROCESS (freq_in, reset)
194      BEGIN
195          IF (reset = '1') THEN
196              state_edge_counter <= wait_for_start_measure;
197              edge_counter <= 0;
198              enable_edge_counter_ack <= '0';
199          ELSIF (rising_edge(freq_in)) THEN
200              CASE state_edge_counter IS
201                  WHEN wait_for_start_measure =>
202                      IF (df_enable_edge_counter_2 = '1') THEN
203                          edge_counter <= 0;
204                          enable_edge_counter_ack <= '1';
205                          state_edge_counter <= counting;
206                      ELSE
207                          state_edge_counter <= wait_for_start_measure;
208                      END IF;
209
210              WHEN counting => IF (df_enable_edge_counter_2 = '0') THEN
211                  enable_edge_counter_ack <= '0';
212                  state_edge_counter <= wait_for_start_measure;
213          ELSE
214              IF (edge_counter < edge_counter_max) THEN
215                  edge_counter <= edge_counter + 1;
216              END IF;
217              state_edge_counter <= counting;
218          END IF;
219          WHEN OTHERS => state_edge_counter <= wait_for_start_measure;
220      END CASE;
221      END IF;
222  END PROCESS;
```

```
--2nd frequency input--
-----------------------
-- FSM for edge counter for 2nd frequency
-- Wait_for_start_measure: Waiting for counting flag to be triggered
-- Counting:    Main edge counting stage, checks ack bit at the same time.
-- If ack is not received, then the edge counter returns to 'wait_for_start_measure' stage
L_EDGE_COUNTER_2 : PROCESS (freq_in_2, reset)
    BEGIN
        IF (reset = '1') THEN
            state_edge_counter_2 <= wait_for_start_measure;
            edge_counter_2 <= 0;
            enable_edge_counter_ack_2 <= '0';
        ELSIF (rising_edge(freq_in_2)) THEN
            CASE state_edge_counter_2 IS
                WHEN wait_for_start_measure =>
                IF (df_enable_edge_counter_2_2 = '1') THEN
                    edge_counter_2 <= 0;
                    enable_edge_counter_ack_2 <= '1';
                    state_edge_counter_2 <= counting;
                ELSE
                state_edge_counter_2 <= wait_for_start_measure;
            END IF;

            WHEN counting =>
            IF (df_enable_edge_counter_2_2 = '0') THEN
            enable_edge_counter_ack_2 <= '0';
            state_edge_counter_2 <= wait_for_start_measure;
        ELSE
            IF (edge_counter_2 < edge_counter_max) THEN
            edge_counter_2 <= edge_counter_2 + 1;
            END IF;
            state_edge_counter_2 <= counting;
        END IF;
        WHEN OTHERS => state_edge_counter_2 <= wait_for_start_measure;
    END CASE;
    END IF;
END PROCESS;
```

```
464  --3rd frequency input--
465  ----------------------
466  -- FSM for edge counter for 3rd frequency
467  -- Wait_for_start_measure: Waiting for counting flag to be triggered
468  -- Counting: Main edge counting stage, checks ack bit at the same time.
469  -- If ack is not received, then the edge counter returns to 'wait_for_start_measure' stage
470  L_EDGE_COUNTER_3 : PROCESS (freq_in_3, reset)
471      BEGIN
472          IF (reset = '1') THEN
473              state_edge_counter_3 <= wait_for_start_measure;
474              edge_counter_3 <= 0;
475              enable_edge_counter_ack_3 <= '0';
476          ELSIF (rising_edge(freq_in_3)) THEN
477              CASE state_edge_counter_3 IS
478                  WHEN wait_for_start_measure =>
479                  IF (df_enable_edge_counter_2_3 = '1') THEN
480                      edge_counter_3 <= 0;
481                      enable_edge_counter_ack_3 <= '1';
482                      state_edge_counter_3 <= counting;
483                  ELSE
484                      state_edge_counter_3 <= wait_for_start_measure;
485              END IF;
486
487              WHEN counting => IF (df_enable_edge_counter_2 = '0') THEN
488              enable_edge_counter_ack_3 <= '0';
489              state_edge_counter_3 <= wait_for_start_measure;
490          ELSE
491              IF (edge_counter_3 < edge_counter_max) THEN
492                  edge_counter_3 <= edge_counter_3 + 1;
493              END IF;
494              state_edge_counter_3 <= counting;
495          END IF;
496          WHEN OTHERS => state_edge_counter_3 <= wait_for_start_measure;
497      END CASE;
498      END IF;
499  END PROCESS;
```

147.7. Time measurement process

The codes for time measurement for the three frequency subsystems are given below in [FIGURE]. As mentioned earlier, the state machine for edge counting consists of four states: Wait_for_enable_measure, Wait_for_ack_1, Measure_timing and Wait_for_ack_2. The major task of this state machine is to control the edge counting process by enabling the edge counting and at the same time track the timing of that window, defined as 60000. Another important task of this state machine is that when the edge counting process stops, then send_trigger is activated [LINE], which enables the transmitter of the UART to initiate the transmission of the counter data (L_Transmitter, will be discussed later).

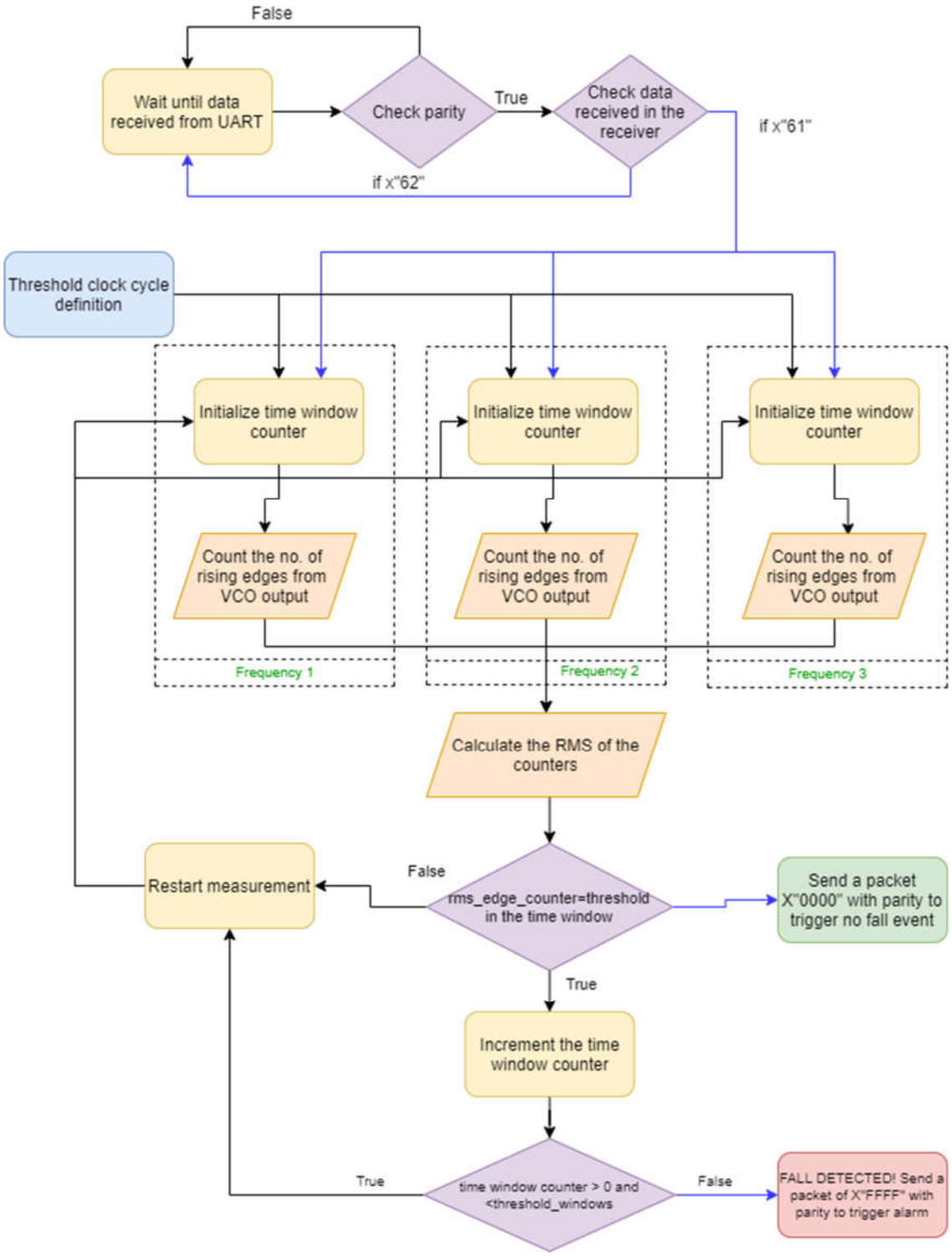
```
L_MEASURE_TIMER : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        state_measure_timer <= wait_for_enable_measure;
        enable_edge_counter <= '0';
        send_trigger <= '0';
        time_counter <= 0;
        new_data <= (OTHERS => '0');
    ELSIF (rising_edge(clk)) THEN
        CASE state_measure_timer IS
            WHEN wait_for_enable_measure =>  send_trigger <= '0'; IF (enable_measure = '1') THEN
                    enable_edge_counter <= '1';
                    state_measure_timer <= wait_for_ack_1;
                ELSE
                    state_measure_timer <= wait_for_enable_measure;
                END IF;

            WHEN wait_for_ack_1 => IF (df_enable_edge_counter_ack_2 = '1' ) THEN state_measure_timer <= measure_timing;
                ELSE
                    state_measure_timer <= wait_for_ack_1;
                END IF;
            WHEN measure_timing => IF (time_counter = time_counter_max) THEN time_counter <= 0;
                    enable_edge_counter <= '0';
                    state_measure_timer <= wait_for_ack_2;
                    counter_ended <= '1';
                ELSE
                    time_counter <= time_counter + 1;
                    state_measure_timer <= measure_timing;
                    counter_ended <= '0';
                END IF;

            WHEN wait_for_ack_2 => counter_ended <= '0'; IF (df_enable_edge_counter_ack_2 = '0') THEN
                    send_trigger <= '1'; --------------------------
                    IF (fall_adl_detected = "00") THEN
                        new_data <= (OTHERS => '0');
                    ELSIF (fall_adl_detected = "01") THEN
                        new_data <= (OTHERS => '0');
                    ELSE
                        new_data <= (OTHERS => '1');
                    END IF;
                    state_measure_timer <= wait_for_enable_measure;

                ELSE
                    state_measure_timer <= wait_for_ack_2;
                END IF;
            WHEN OTHERS =>
                state_measure_timer <= wait_for_enable_measure;
        END CASE;
    END IF;
END PROCESS;
```

```
L_MEASURE_TIMER_2 : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        state_measure_timer_2 <= wait_for_enable_measure;
        enable_edge_counter_2 <= '0';
        send_trigger_2 <= '0';
        time_counter_2 <= 0;
        new_data_2 <= (OTHERS => '0');
    ELSIF (rising_edge(clk)) THEN
        CASE state_measure_timer_2 IS
            WHEN wait_for_enable_measure =>                send_trigger_2 <= '0';
                IF (enable_measure_2 = '1') THEN                    enable_edge_counter_2 <= '1';
                    state_measure_timer_2 <= wait_for_ack_1;
                ELSE
                    state_measure_timer_2 <= wait_for_enable_measure;
                END IF;

            WHEN wait_for_ack_1 =>  IF (df_enable_edge_counter_ack_2_2 = '1' ) THEN  state_measure_timer_2 <= measure_timing;
                ELSE
                    state_measure_timer_2 <= wait_for_ack_1;
                END IF;

            WHEN measure_timing => IF (time_counter_2 = time_counter_max) THEN time_counter_2 <= 0;
                    enable_edge_counter_2 <= '0';
                    state_measure_timer_2 <= wait_for_ack_2;
                    counter_ended_2 <= '1';
                ELSE
                    time_counter_2 <= time_counter_2 + 1;
                    state_measure_timer_2 <= measure_timing;
                    counter_ended_2 <= '0';
                END IF;

            WHEN wait_for_ack_2 =>  counter_ended_2 <= '0';   IF (df_enable_edge_counter_ack_2_2 = '0') THEN send_trigger_2 <= '1'; -------------------------
                    IF (fall_adl_detected = "00") THEN
                        new_data_2 <= (OTHERS => '0');
                    ELSIF (fall_adl_detected = "01") THEN
                        new_data_2 <= (OTHERS => '0');
                    ELSE
                        new_data_2 <= (OTHERS => '1');
                    END IF;
                    state_measure_timer_2 <= wait_for_enable_measure;

                ELSE
                    state_measure_timer_2 <= wait_for_ack_2;
                END IF;
            WHEN OTHERS =>
                state_measure_timer_2 <= wait_for_enable_measure;
        END CASE;
    END IF;
END PROCESS;
```

```
L_MEASURE_TIMER_3 : PROCESS (clk, reset)
BEGIN
    IF (reset = '1') THEN
        state_measure_timer_3 <= wait_for_enable_measure;
        enable_edge_counter_3 <= '0';

        send_trigger_3 <= '0';
        time_counter_3 <= 0;
        new_data_3 <= (OTHERS => '0');
    ELSIF (rising_edge(clk)) THEN
        CASE state_measure_timer_3 IS
            WHEN wait_for_enable_measure => send_trigger_3 <= '0'; IF (enable_measure_3 = '1') THEN
                    enable_edge_counter_3 <= '1';
                    state_measure_timer_3 <= wait_for_ack_1;
                ELSE
                    state_measure_timer_3 <= wait_for_enable_measure;
                END IF;

            WHEN wait_for_ack_1 =>  IF (df_enable_edge_counter_ack_2_3 = '1' ) THEN
                    state_measure_timer_3 <= measure_timing;
                ELSE
                    state_measure_timer_3 <= wait_for_ack_1;
                END IF;
            WHEN measure_timing =>  IF (time_counter_3 = time_counter_max) THEN
                    time_counter_3 <= 0;
                    enable_edge_counter_3 <= '0';
                    state_measure_timer_3 <= wait_for_ack_2;
                    counter_ended_3 <= '1';
                ELSE
                    time_counter_3 <= time_counter + 1;
                    state_measure_timer_3 <= measure_timing;
                    counter_ended_3 <= '0';
                END IF;
            WHEN wait_for_ack_2 =>
                counter_ended_3 <= '0';
                IF (df_enable_edge_counter_ack_2_3 = '0') THEN send_trigger_3 <= '1'; IF (fall_adl_detected = "00") THEN
                        new_data_3 <= (OTHERS => '0');
                    ELSIF (fall_adl_detected = "01") THEN
                        new_data_3 <= (OTHERS => '0');
                    ELSE
                        new_data_3 <= (OTHERS => '1');
                    END IF;
                    state_measure_timer_3 <= wait_for_enable_measure;

                ELSE
                    state_measure_timer_3 <= wait_for_ack_2;
                END IF;
            WHEN OTHERS =>
                state_measure_timer_3 <= wait_for_enable_measure;
        END CASE;
    END IF;
END PROCESS;
```

157.7. System Architecture

15.17.7.1 Block diagram



[FIGURE BLOCK DIAGRAM SHOULD BE IMPLEMENTED IN THIS SECTION TOMORROW]

15.27.7.2 Working Principle

The algorithm for the three-channel frequency measuring and interrupt generation based on fall/non-fall event is shown in [FIGURE]. Data stream is initially received through UART to activate or deactivate the frequency monitoring and the sub-block for interrupt generation. A parity bit which was sent with the data for error checking is checked at first, and if the underlying error checking definition returns true, then the data packets are checked. Two-bit packet flags determine whether the measurement process should start or not:

- X”61”: the frequency monitoring for the three-channels and interrupt sub-blocks are triggered to start working.
- X”62”: No action will go on; measurement will stop and UART will wait for the other packets on the Rx channel.

The three-frequency measuring sub-blocks perform similar operations. The definition for thresholds is assigned as a common resource for all them. The defined number of clock cycles for threshold frequency was set as 10 MHz, corresponding to the max. acceleration derived in section 5.9.
When the frequency measuring block starts, a time window of 0.5 ms is created, and the rising edges from the three incoming signals from the three VCOs are counted using three separate counters. The counting process continues until the end of the window.
The following operations occur after the edge counting and timing measurement.

- The RMS value of the three edge counters is calculated, and this rms_edge_counter is compared with our defined threshold.
- If the rms_edge_counter is equal or higher than the threhsold, then the time_window counter is incremented by 1, a packet of X”0000” with parity bit is sent to not trigger any fall alarm. The system returns to the three-frequency measuring state.
- If the time window counter is greater than 0, but less than threshold_windows, then fall_adl_detected signal becomes “10”, which is the corresponding code for ADL of our system.
- If the time window counter equals to threshold_windows (which is 5 in this case, i.e. 11.82 ms), and the rms_edge_counter is less than or equal to the threshold counter, a fall event is confirmed, and a trigger is sent to activate the alarm by sending a packet of X”FFFF”.

167.8. RTL Model Complementation

16.1 7.8.1. Interrupt generation

The resources and the VHDL processes for the time window counter and interrupt generation are shown in figure [FIGURE] and [FIGURE]. As mentioned before, the rising edges of the three incoming signals from three VCOs are counted using three separate counters in 0.5 ms time windows. The three counting processes work concurrently, and the frequency is measured until the end of the window.

```
4   ----------------------------------------------------------------------------------------------
5   ENTITY FMEAS_UART_ANZEN IS
6       GENERIC (
7           -- maximum frequency in our case- 25MHz- the
8           -- maximum value of the counter triggered be the measured frequency input:
9           -- def.: 59100: timewindow*freq_in_max = 0.5*25 MHz = 12500
10          edge_counter_max : INTEGER := 12500;   --1ms
11          -- the maximum value of the counter triggered by the clock input (sample time definition):
12          -- def.: 283680: timewindow*clockfrequency = 0.5 * 120 MHz = 60000
13          time_counter_max : INTEGER := 60000; --1ms
14          ---- the amount of clock cycles (clk) occuring during a single baud (UART speed definition): def.: 3125: 38400 bps
15          -- ( if you have 38400 BAUD rate, 38400 bits/sec, you have 120 M cycles in one sec, you will need 120M/38400= cycles for 1 bit
16          baud_pulses : INTEGER := 3125;
17          -- --we trigger if the frequency value is higher than 10MHz ; 10 * 0.5 = 5000
18          triggering_level : INTEGER := 5000;
19          -- trigger if the value within the range is found for 5 time windows
20          time_windows : INTEGER := 5);
21      PORT (
22          clk : IN STD_LOGIC;
23          reset : IN STD_LOGIC;
24          freq_in,freq_in_2,freq_in_3 : IN STD_LOGIC; --3 frequency inputs from 3 VCOs
25          rx : IN STD_LOGIC;
26          tx : OUT STD_LOGIC);
27  END FMEAS_UART_ANZEN;
28  ----------------------------------------------------------------------------------------------
29

144     ----------------------------------------------------------------------------------------------
145     -- time window and interrupt generation resources ------------------------------------------------------------------
146     SIGNAL counter_ended,counter_ended_2,counter_ended_3 : STD_LOGIC := '0'; --counter status flags for three frequencies
147     SIGNAL time_window_counter : INTEGER RANGE 0 TO time_windows := 0; --signal to keep track of the current counted time windows
148     SIGNAL fall_adl_detected : STD_LOGIC_VECTOR (1 DOWNTO 0) := "00"; --00: no event, 10: ADL-not fall, 11: fall
149     CONSTANT threshold_frequency : INTEGER := triggering_level; --if equal to value of 10MHz
150   SIGNAL RMS_EDGE_COUNTER : INTEGER :=0;
151     ----------------------------------------------------------------------------------------------
```

Figure 26 the upper two makes one figurer

The discussions regarding the explanation of the code given below have been mentioned in the previous section. In addition to that we can see when a fall event is not triggered, a packet of X”0000” is sent through UART, and the time window counter gets reset. The triggering action is considered for both the time counter and low input frequency cases.

```
600   ------------------------------------------------
601   -------TIME WINDOW COUNTER FOR FALL AND ADL----
602
603   TIME_WINDOW_COUNTER_FALL_ADL : PROCESS (clk, reset, time_window_counter)
604   VARIABLE temp_a : unsigned(31 downto 0) :="00000000000000000000000000110010";   --50
605   VARIABLE temp_b : unsigned(15 downto 0) :=(others => '0');
606   VARIABLE temp_c : INTEGER :=0;
607   BEGIN
608       IF (reset = '1') THEN
609           time_window_counter <= 0;
610           fall_adl_detected <= "00";
611           RMS_EDGE_COUNTER <=0;
612       ELSIF (rising_edge(clk)) THEN
613           IF (counter_ended = '1' and counter_ended_2 = '1' and counter_ended_3 = '1') THEN
614               temp_c := (edge_counter*edge_counter)+(edge_counter_2*edge_counter_2)+(edge_counter_3*edge_counter_3);
615
616               temp_a := to_unsigned(temp_c,temp_a'length);
617               temp_b := sqrt ( temp_a );
618               RMS_EDGE_COUNTER <= to_integer(temp_b);
619
620               IF (RMS_EDGE_COUNTER >= triggering_level) THEN ----- dont trigger
621                   time_window_counter <= 0;
622                   fall_adl_detected <= "00";
623               ELSIF (time_window_counter < time_windows) THEN ---- it is less than 10 MHz, initiate the the time window counter
624                   time_window_counter <= time_window_counter + 1;
625                   IF (time_window_counter < (time_windows - 1)) THEN
626                       fall_adl_detected <= "01"; -- some kind of ADL
627                   ELSIF (time_window_counter = (time_windows - 1)) THEN
628                       fall_adl_detected <= "11";
629
630                   END IF;
631               ELSE
632                   time_window_counter <= time_window_counter;
633                   fall_adl_detected <= "11";
634               END IF;
635           ELSE
636               time_window_counter <= time_window_counter;
637           END IF;
638       ELSE
639           time_window_counter <= time_window_counter;
640       END IF;
641   END PROCESS;
```

Figure 27 the main thing. add a good caption

```
151     ---------------------------------------------------------------------------------------------
152     function  sqrt  ( d : UNSIGNED ) return UNSIGNED is
153         variable a : unsigned(31 downto 0):=d;  --original input.
154         variable q : unsigned(15 downto 0):=(others => '0');  --result.
155         variable left,right,r : unsigned(17 downto 0):=(others => '0');  --input to adder/sub.r-remainder.
156         variable i : integer:=0;
157
158         begin
159         for i in 0 to 15 loop
160         right(0):='1';
161         right(1):=r(17);
162         right(17 downto 2):=q;
163         left(1 downto 0):=a(31 downto 30);
164         left(17 downto 2):=r(15 downto 0);
165         a(31 downto 2):=a(29 downto 0);  --shifting by 2 bit.
166         if ( r(17) = '1') then
167         r := left + right;
168         else
169         r := left - right;
170         end if;
171         q(15 downto 1) := q(14 downto 0);
172         q(0) := not r(17);
173         end loop;
174         return q;
175
176     end sqrt;
```

Figure 28 to generate square root [citation required, can you find Pallavi citations for square root gen in vhdl plz]

16.2 7.8.2. UART transmitter

The transmitter FSM has six states. The details of the transmission is given below in Table [TABLE]. The code for the UART transmitter is given in Figure [FIGURE].

States	Description
wait_for_send_trigger	Waits until the next send request is received. There are three send triggers for the three frequencies, which becomes '1' by the end of the corresponding time windows of the three inputs. If a fall is detected (true event) for the time window threshold (5 in our case), then X"FFFF" packet is transmitted, else X"0000" is sent at the end of each of the time windows.
send_start_bit	Checks if the tx_counter is equal to the baud pulse rate of the system. If true, then start sending the data packets, and restart the tx_counter. Else, increment the tx_counter until it becomes equal to the baud pulse rate.
send_packets	The first start bit is sent to the processor to inform the data incoming. The first packet of X"FF" is transmitted.
set_bit_counter	The bit counter transmits the 8-bits via through its definition as a STD_LOGIC_VECTOR. The parity bit state comes next when the first packet is transmitted.
send_parity_bit	Parity bit is generated using the XOR gate and sent.
send_stop_bits	Stop bit is transmitted indicating the end of transmission

```
642  ----------------------------------------------------------------------------------------------
643  ----------------------------------TRANSMITTER SECTION-----------------------------------------
644  ----------------------------------------------------------------------------------------------
645  L_TRANSMITTER : PROCESS (clk, reset)
646  BEGIN
647      IF (reset = '1') THEN
648          state_transmitter <= wait_for_send_trigger;
649          packet_0 <= (OTHERS => '0');
650          packet_1 <= (OTHERS => '0');
651          tx_counter <= 0;
652          bit_counter <= 0;
653          packet_counter <= 0;
654          tx <= '1';
655      ELSIF (rising_edge(clk)) THEN
656          CASE state_transmitter IS
657              WHEN wait_for_send_trigger =>
658              IF (send_trigger = '1') THEN --1st frequency send trigger checking and transmission initiation
659                  packet_0 <= STD_LOGIC_VECTOR(new_data(7 DOWNTO 0));
660                  packet_1 <= STD_LOGIC_VECTOR(new_data(15 DOWNTO 8));
661                  state_transmitter <= send_start_bit;
662              ELSIF (send_trigger_2 = '1') THEN --2nd frequency send trigger checking and transmission initiation
663                  packet_0 <= STD_LOGIC_VECTOR(new_data_2(7 DOWNTO 0));
664                  packet_1 <= STD_LOGIC_VECTOR(new_data_2(15 DOWNTO 8));
665                  state_transmitter <= send_start_bit;
666              ELSIF (send_trigger_3 = '1') THEN --3rd frequency send trigger checking and transmission initiation
667                  packet_0 <= STD_LOGIC_VECTOR(new_data_3(7 DOWNTO 0));
668                  packet_1 <= STD_LOGIC_VECTOR(new_data_3(15 DOWNTO 8));
669                  state_transmitter <= send_start_bit;
670              ELSE
671                  state_transmitter <= wait_for_send_trigger;
672          END IF;
673
674          WHEN send_start_bit => tx <= '0';
675          IF (tx_counter = baud_pulses) THEN
676              tx_counter <= 0;
677              state_transmitter <= send_packets;
678          ELSE
679              tx_counter <= tx_counter + 1;
680              state_transmitter <= send_start_bit;
681          END IF;
682          WHEN send_packets => state_transmitter <= send_packets;
683
684              IF (packet_counter = 0) THEN
685                  tx <= packet_0(bit_counter);
686              ELSIF (packet_counter = 1) THEN
```

```
686             ELSIF (packet_counter = 1) THEN
687                 tx <= packet_1(bit_counter);
688             END IF;
689 
690             IF (tx_counter = baud_pulses) THEN
691                 tx_counter <= 0;
692                 state_transmitter <= set_bit_counter;
693             ELSE
694                 tx_counter <= tx_counter + 1;
695             END IF;
696 
697         WHEN set_bit_counter => IF (bit_counter = 7) THEN
698         bit_counter <= 0;
699         state_transmitter <= send_parity_bit;
700     ELSE
701         bit_counter <= bit_counter + 1;
702         state_transmitter <= send_packets;
703     END IF;
704     WHEN send_parity_bit => tx <= ((packet_1(7) XOR packet_1 (6)) XOR (packet_1(5) XOR packet_1 (4)))
705     XOR ((packet_1(3) XOR packet_1 (2)) XOR (packet_1(1) XOR packet_1 (0)))
706     XOR ((packet_0(7) XOR packet_0 (6)) XOR (packet_0(5) XOR packet_0(4)))
707     XOR ((packet_0(3) XOR packet_0(2)) XOR (packet_0(1) XOR packet_0(0)));
708 
709     IF (tx_counter = baud_pulses) THEN
710         tx_counter <= 0;
711         state_transmitter <= send_stop_bits;
712     ELSE
713         tx_counter <= tx_counter + 1;
714         state_transmitter <= send_parity_bit;
715     END IF;
716     WHEN send_stop_bits => tx <= '1';
717     IF (tx_counter = 2 * baud_pulses) THEN
718     -- IF (tx_counter = 4 * baud_pulses) THEN
719         tx_counter <= 0;
720 
721         IF (packet_counter = 1) THEN
722             packet_counter <= 0;
723             state_transmitter <= wait_for_send_trigger;
724         ELSE
725             packet_counter <= packet_counter + 1;
726             state_transmitter <= send_start_bit;
727         END IF;
728 
729       ELSE
730           tx_counter <= tx_counter + 1;
731       END IF;
732       WHEN OTHERS => state_transmitter <= wait_for_send_trigger;
733   END CASE;
734   END IF;
735   END PROCESS;
```

16.3 7.8.3. UART receiver

The UART receiver receives signal the Rx line of the UART. The receiver FSM consists of seven states. The description of the operation of the receiver FSM is given below in Table [TABLE]. The code for the receiver module is given in Figure [FIGURE].

States	Description
wait_for_start_bit	If negative edge rx flag is enabled, then start bit delay state. Else, wait until the flag turns '0'.
start_bit_delay	Wait for 1.5 bauds before sampling. No autobaud rate capability, communication is expected at 38400 baud pulse. If rx_counter reaches 1.5 bauds, then start sampling, else increment the counter and retain the state.
sample_bit	Store the RX values in the "received_data" buffer, the indexing is controlled using the rx_bit_counter. If the rx_bit_counter is not 7, then increment the counter and go to the delay_bit state. This corresponds to the delay bit in the received data. Else if rx_counter is 7, then all the bits have been received, the system moves on to check the parity.
delay_bit	Increment the rx_counter until it is equal to the baud_pulse, and return to the sample_bit state.
wait_for_parity_bit	Receive the parity bit if rx_counter matches the baud pulses, else wait until parity bit is received.
delay_stop_bits	Increment the rx_counter until it is equal to twice the baud_pulses in order to start decoding, else increment rx_counter and retain the state.
decode_received_data	The parity is checked at the beginning of the state for any error, if found then the system returns to the first state. Else the received_data is checked whether it is and enable type or disable type. If enable, then it enables all the enable measure triggers for the three frequencies. If disable, then it does the disable operation.

```
742   --RX shift register-----------------------------------------------------------------------------
743   -- Used to store the current and previous RX states. Used by RX_EDGE_DETECTION to determine
744   -- the presence of a falling edge in the RX input
745   ------------------------------------------------------------------------------------------------
746   INPUT_SHREGS : PROCESS (clk, reset)
747   BEGIN
748       IF (reset = '1') THEN
749           sh_rx <= "11";
750       ELSIF (rising_edge(clk)) THEN
751           sh_rx(1) <= sh_rx(0);
752           sh_rx(0) <= rx;
753       END IF;
754   END PROCESS;
755
756   --Falling edge detection of RX input -------------------------------------------------------------
757   -- The presence of a falling edge is used as an indicator that data is incoming
758   ------------------------------------------------------------------------------------------------
759   RX_EDGE_DETECTION : PROCESS (clk, reset)
760   BEGIN
761       IF (reset = '1') THEN
762           rx_d <= '1';
763       ELSIF (rising_edge(clk)) THEN
764           rx_d <= sh_rx(1);
765       END IF;
766   END PROCESS;
767
768   negedge_rx <= (NOT sh_rx(1)) AND rx_d;
```

```
774  L_RECEIVER : PROCESS (clk, reset)
775  BEGIN
776      IF (reset = '1') THEN
777          state_receiver <= wait_for_start_bit;
778          rx_counter <= 0;
779          bit_counter_rx <= 0;
780          received_data <= (OTHERS => '0');
781          enable_measure <= '0';
782          enable_measure_2 <= '0';
783          enable_measure_3 <= '0';
784      ELSIF (rising_edge(clk)) THEN
785          CASE state_receiver IS
786          WHEN wait_for_start_bit =>
787              IF (negedge_rx = '1') THEN
788                  state_receiver <= start_bit_delay;
789              ELSE
790                  state_receiver <= wait_for_start_bit;
791              END IF;
792          WHEN start_bit_delay =>
793              IF (rx_counter = (3 * baud_pulses)/2) THEN
794              -- IF (rx_counter = (4 * baud_pulses)) THEN
795                  rx_counter <= 0;
796                  state_receiver <= sample_bit;
797              ELSE
798                  rx_counter <= rx_counter + 1;
799                  state_receiver <= start_bit_delay;
800              END IF;
801          WHEN sample_bit => received_data(bit_counter_rx) <= rx_d;
802              IF (bit_counter_rx = 7) THEN
803                  bit_counter_rx <= 0;
804                  state_receiver <= wait_for_parity_bit;
805
806              ELSE
807                  bit_counter_rx <= bit_counter_rx + 1;
808                  state_receiver <= delay_bit;
809              END IF;
810          WHEN delay_bit =>
811              IF (rx_counter = baud_pulses) THEN
812                  rx_counter <= 0;
813                  state_receiver <= sample_bit;
814              ELSE
815                  rx_counter <= rx_counter + 1;
816                  state_receiver <= delay_bit;
```

```
817             END IF;
818         WHEN wait_for_parity_bit =>
819             IF (rx_counter = baud_pulses) THEN
820                 rx_counter <= 0;
821                 parity_bit_checker <= rx_d;
822                 state_receiver <= delay_stop_bits;
823             ELSE
824                 rx_counter <= rx_counter + 1;
825                 state_receiver <= wait_for_parity_bit;
826             END IF;
827 
828         WHEN delay_stop_bits =>
829             IF (rx_counter = 2 * baud_pulses) THEN
830             -- IF (rx_counter = 4 * baud_pulses) THEN
831             rx_counter <= 0;
832             state_receiver <= decode_received_data;
833             ELSE
834             rx_counter <= rx_counter + 1;
835             state_receiver <= delay_stop_bits;
836             END IF;
837 
838         WHEN decode_received_data =>
839         IF ((((received_data(7)XOR received_data(6)) XOR (received_data(5)XOR received_data(4)))
840         XOR ((received_data(3)XOR received_data(2))
841         XOR (received_data(1)XOR received_data(0)))) /= parity_bit_checker) THEN
842             state_receiver <= wait_for_start_bit;
843         ELSE
844             CASE received_data IS
845                 WHEN ctrl_enable_measure =>
846                     enable_measure <= '1'; enable_measure_2 <= '1'; enable_measure_3 <= '1';
847                 WHEN ctrl_disable_measure =>
848                     enable_measure <= '0'; enable_measure_2 <= '0'; enable_measure_3 <= '0';
849                 WHEN OTHERS => state_receiver <= wait_for_start_bit;
850             END CASE;
851         END IF;
852         state_receiver <= wait_for_start_bit;
853         WHEN OTHERS => state_receiver <= wait_for_start_bit;
854 END CASE;
855 END IF;
856 END PROCESS;
857 END RTL_FMEAS_UART_ANZEN;
```

177.9. Functional model simulation

A testbench was programmed for our VHDL code and was verified. This section discusses the testbench setup, parameter choices and the corresponding simulations performed via ModelSim.

17.1 7.9.1. Test bench

The DUT model was instantiated inside the testbench using the following declarations of the architecture RTL_FMEAS_UART_ANZEN (Figure [FIGURE]).

```
7   architecture RTL_FMEAS_UART_ANZEN of testbench is
8
9       constant     clock_cycle:       time := 8.333 ns;       -- 120 MHz
10      signal        freq_in_cycle:    time := 66.67 ns;            -- 15 MHz
11      signal        freq_in_cycle_2:  time := 83.33 ns;            -- 12 MHz
12      signal        freq_in_cycle_3:  time := 52.63 ns;            -- 19 MHz
13
14      signal clk:                   std_logic := '0';
15      signal reset:                 std_logic := '1';
16      signal freq_in:               std_logic := '0';
17      signal freq_in_2:             std_logic := '0';
18      signal freq_in_3:             std_logic := '0';
19      signal tx:                    std_logic;
20      signal rx:                    std_logic := '1';
21
22      constant c_BIT_PERIOD : time := 26041.66667 ns; --defined by our baud rate; 1/38400 bps = 26041.66667 ns
23      constant twice_c_BIT_PERIOD : time := 52083.33334 ns;
```

The DUT was also tested for different random frequencies for the 3 frequency inputs. The choice of the random stimuli was made to analyze the behavior of the system. Figure [FIGURE] shows the frequency stimuli process. Among the 6, 2 of the cases corresponds to possible fall cases.

```
L_FREQ_IN_CYCLE: process
begin
    wait for 430 ns;
    reset <= '0';
    wait for 12 ms;

    freq_in_cycle <= 77 ns; freq_in_cycle_2 <= 57 ns; freq_in_cycle_3 <= 89 ns;  -- freq_1=12.98 Mhz;  freq_2=17.54 Mhz; freq_3=11.26 Mhz
    wait for 5 ms;

    freq_in_cycle <= 62 ns;  freq_in_cycle_2 <= 74 ns; freq_in_cycle_3 <= 55 ns;  -- freq_1=15.38 Mhz;  freq_2=13.51 Mhz; freq_3=18.18 Mhz
    wait for 5 ms;

    freq_in_cycle <= 892 ns;  freq_in_cycle_2 <= 178 ns; freq_in_cycle_3 <= 340 ns;  --POSSIBLE FALL CASE. freq_1=10.87 Mhz;  freq_2=5.61 Mhz; freq_3=2.94 Mhz
    wait for 5 ms;

    freq_in_cycle <= 42 ns;  freq_in_cycle_2 <= 51 ns; freq_in_cycle_3 <= 75 ns;  -- freq_1=23.81 Mhz;  freq_2=19.60 Mhz; freq_3=13.33 Mhz
    wait for 5 ms;

    freq_in_cycle <= 81 ns; freq_in_cycle_2 <= 245 ns; freq_in_cycle_3 <= 67 ns;    -- freq_1=12.35 Mhz;  freq_2=4.081 Mhz; freq_3=14.92 Mhz
    wait for 5 ms;

    freq_in_cycle <= 123 ns; freq_in_cycle_2 <= 167 ns; freq_in_cycle_3 <= 244 ns;   -- POSSIBLE FALL CASE. freq_1=8.13 Mhz;  freq_2=5.988 Mhz; freq_3=4.098 Mhz
    wait;
end process;
```

Figure[FIGURE] shows the receiver process of the UART receiver. According to our defined system, X"61" corresponded to the start reading operation, while X"62" corresponded to the stop operation. The bit transmission delays testing were performed based on the baud rate of the system, i.e. 38400. This bit period was calculated from the following equation:

$$c_{BIT_{PERIOD}} = \frac{1}{38400}$$

Along with the start measuring and stop measuring packets, we can also see the parity bit was added. Concurrently, the code for the generation of clock and input frequency is shown in Figure [FIGURE].

```
54      --PROGRAMMING AND MEASUREMENT CONTROL----------------------------------------
55      -- Programming and measurement initialization expected time: ~1.25 ms
56      -- Frequency measurement (4 windows): ~4*0.5 ms = 2 ms
57      -- Disable measurement expected time: ~0.3125 ms
58      L_RX: process
59      begin
60          wait for 2.5 ms;
61
62              -- start measure frame (0x61) --
63              rx <= '0';                      -- start bit
64              wait for c_BIT_PERIOD;
65
66              rx <= '1';                                          -- bit 0
67              wait for c_BIT_PERIOD;
68
69              rx <= '0';                                          -- bit 1
70              wait for c_BIT_PERIOD;
71
72              rx <= '0';                                          -- bit 2
73              wait for c_BIT_PERIOD;
74
75              rx <= '0';                                          -- bit 3
76              wait for c_BIT_PERIOD;
77
78              rx <= '0';                                          -- bit 4
79              wait for c_BIT_PERIOD;
80
81              rx <= '1';                                          -- bit 5
82              wait for c_BIT_PERIOD;
83
84              rx <= '1';                                          -- bit 6
85              wait for c_BIT_PERIOD;
86
87              rx <= '0';                                          -- bit 7
88              wait for c_BIT_PERIOD;
89
90              rx <= '1';                                          -- stop bit
91              wait for c_BIT_PERIOD;
92
93              rx <= '1';              -- parity
94              wait for twice_c_BIT_PERIOD;
95
```

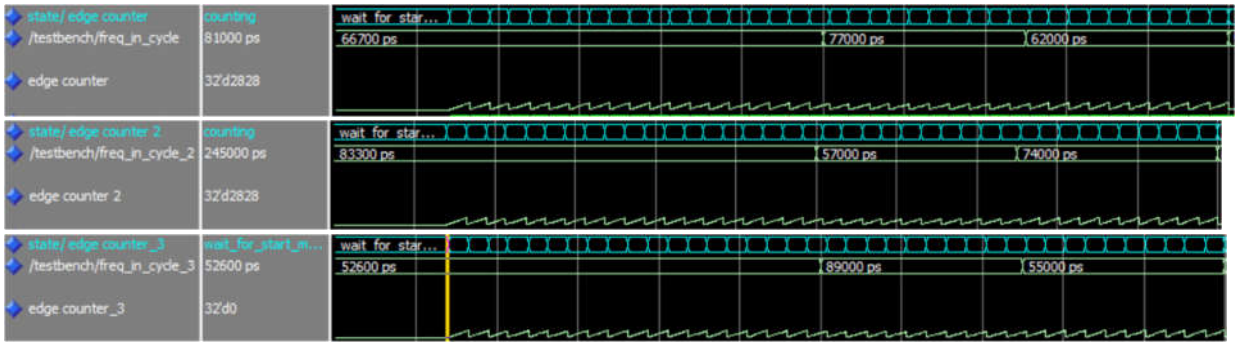
```
97          ---------------
98          wait for 70 ms;
99          ---------------
100
101
102             -- stop measure frame (0x62) --
103             rx <= '0';                      -- start bit
104             wait for c_BIT_PERIOD;
105
106             rx <= '0';                                          -- bit 0
107             wait for c_BIT_PERIOD;
108
109             rx <= '1';                                          -- bit 1
110             wait for c_BIT_PERIOD;
111
112             rx <= '0';                                          -- bit 2
113                 wait for c_BIT_PERIOD;
114
115             rx <= '0';                                          -- bit 3
116                 wait for c_BIT_PERIOD;
117
118             rx <= '0';                                          -- bit 4
119                 wait for c_BIT_PERIOD;
120
121             rx <= '1';                                          -- bit 5
122             wait for c_BIT_PERIOD;
123
124             rx <= '1';                                          -- bit 6
125             wait for c_BIT_PERIOD;
126
127             rx <= '0';                                          -- bit 7
128             wait for c_BIT_PERIOD;
129
130             rx <= '1';                                      -- stop bit
131             wait for c_BIT_PERIOD;
132
133             rx <= '1';              -- parity
134             wait for twice_c_BIT_PERIOD;
135
136
220     L_CLOCK: process
221     begin
222         wait for clock_cycle/2;
223         clk <= not clk;
224     end process;
225
226     L_FREQ_IN: process
227     begin
228         wait for freq_in_cycle/2;
229         freq_in <= not freq_in; freq_in_2 <= not freq_in_2; freq_in_3 <= not freq_in_3;
230     end process;
```

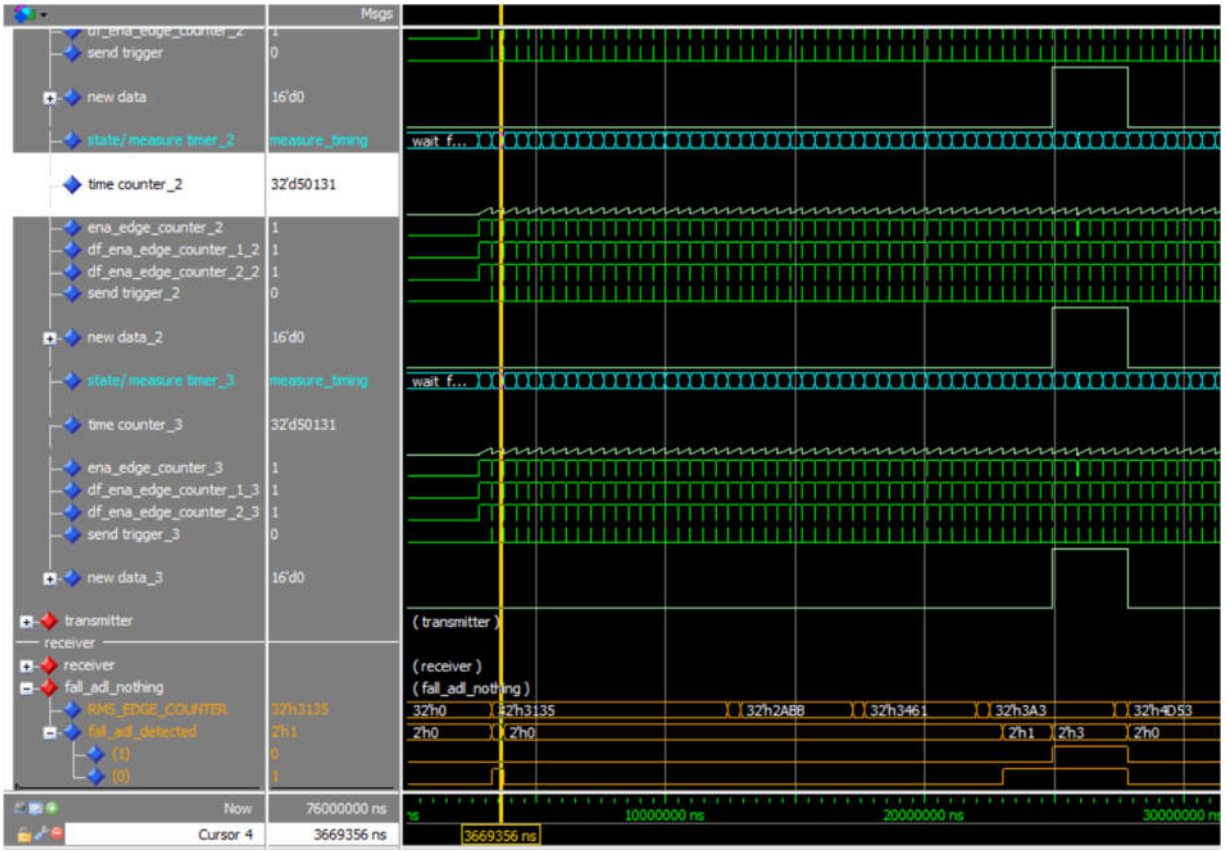
17.2 7.9.3. Testbench Simulation results

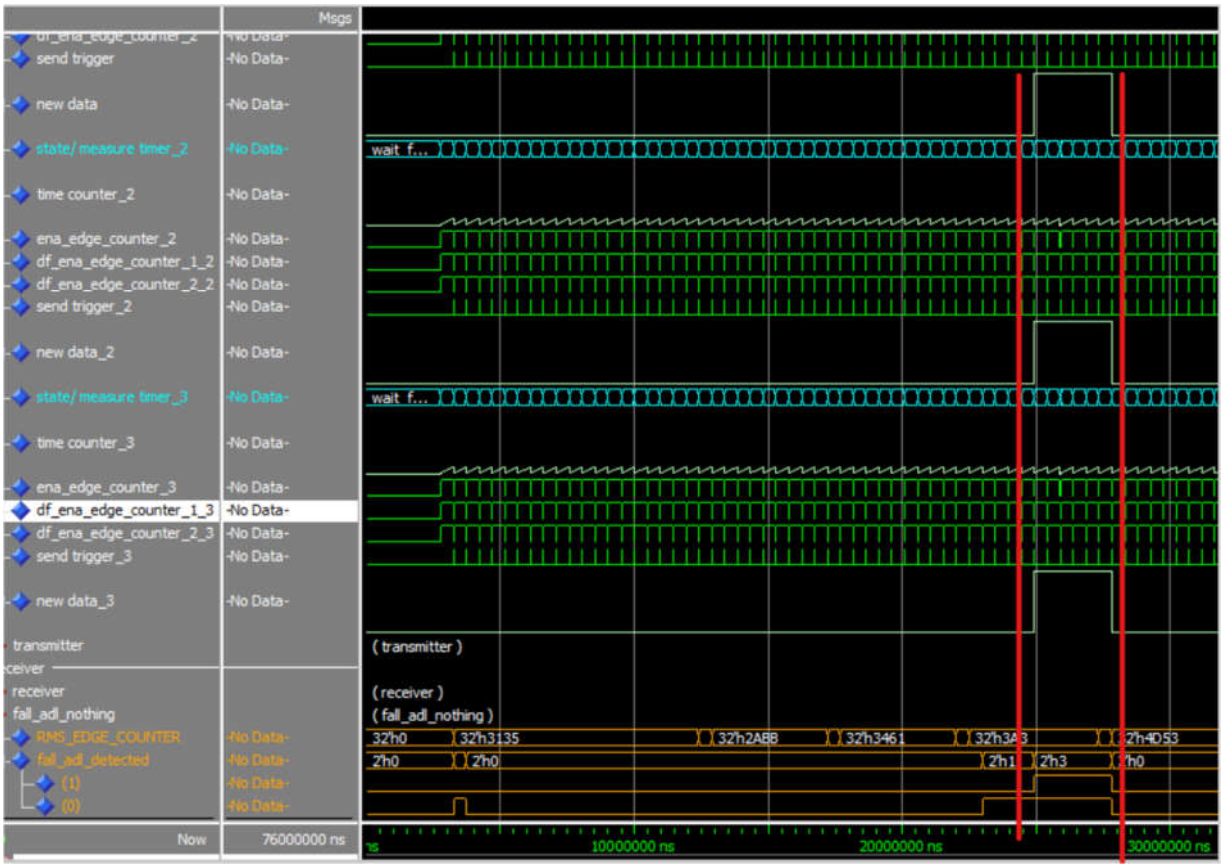
17.2.1 7.3.2.1 Generating interrupts

The DUT in the testbench was tested and the results were observed in the waveform generator of ModelSim. [FIGURE] to [FIGURE] shows the process of characterizing an ADL event and a fall event. A fall is detected when the three input frequencies are less than 10 Mhz, and the resulting rms_edge_counter is less than the threshold triggering level in the defined time window size, for 5 consecutive windows. But if the event does not occur for 5 time windows then it is not characterized as fall, and an ADL code is passed to “fall_adl_detected”.

ADL event is encoded as “01”, fall event as “11”, while a no action event as “00”. When a fall is detected, a new data is assigned X”FFFF” [FIGURE], which is then used for the transmission.




Msgs
send trigger
0
new data
16'd0
state/measure timer_2
measure_timing
wait f...
time counter_2
32'd50131
ena_edge_counter_2
1
df_ena_edge_counter_1_2
1
df_ena_edge_counter_2_2
1
send trigger_2
0
new data_2
16'd0
state/measure timer_3
measure_timing
wait f...
time counter_3
32'd50131
ena_edge_counter_3
1
df_ena_edge_counter_1_3
1
df_ena_edge_counter_2_3
1
send trigger_3
0
new data_3
16'd0
transmitter
(transmitter)
receiver
receiver
(receiver)
fall_adl_nothing
(fall_adl_nothing)
RMS_EDGE_COUNTER
32'h3135
32'h0
32'h3135
32'h2ABB
32'h3461
32'h3A3
32'h4D53
fall_adl_detected
2'h1
2'h0
2'h0
2'h1
2'h3
2'h0
(1)
0
(0)
1
Now
76000000 ns
Cursor 4
3669356 ns
10000000 ns
20000000 ns
30000000 ns
3669356 ns



17.2.2 7.3.2.1 UART Transmitter verification

When a fall happened, a packet of X”FFFF” is sent to the Rx of the UART of the DUT, along with the party bit, otherwise a packet of X”0000” is sent [FIGURE]. Parity was included in the design to detect data errors, and thus increasing the data accuracy. For the parity implementation, even parity was used, thus the parity bit became zero after X”FF” byte [FIGURE].

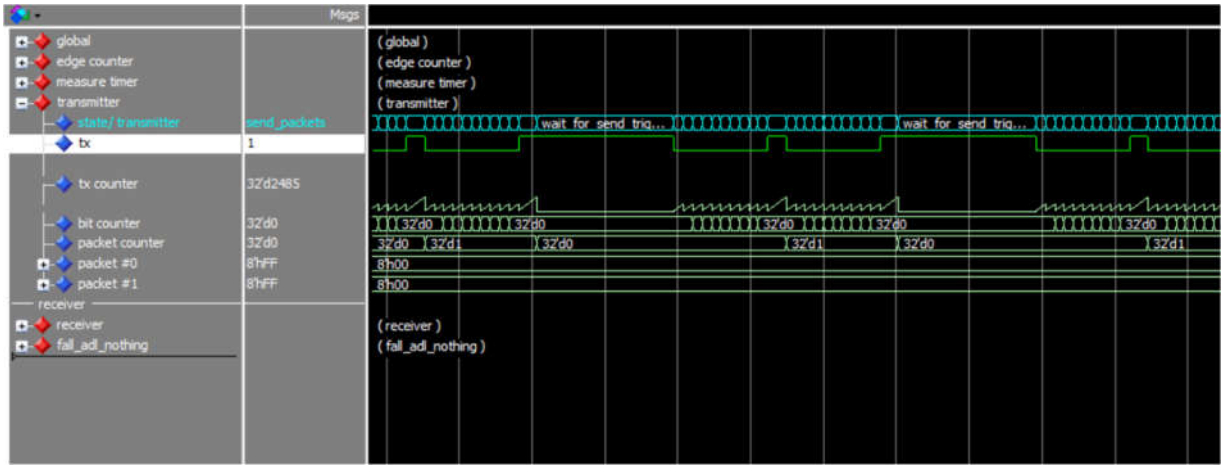


Figure 29 parity figure

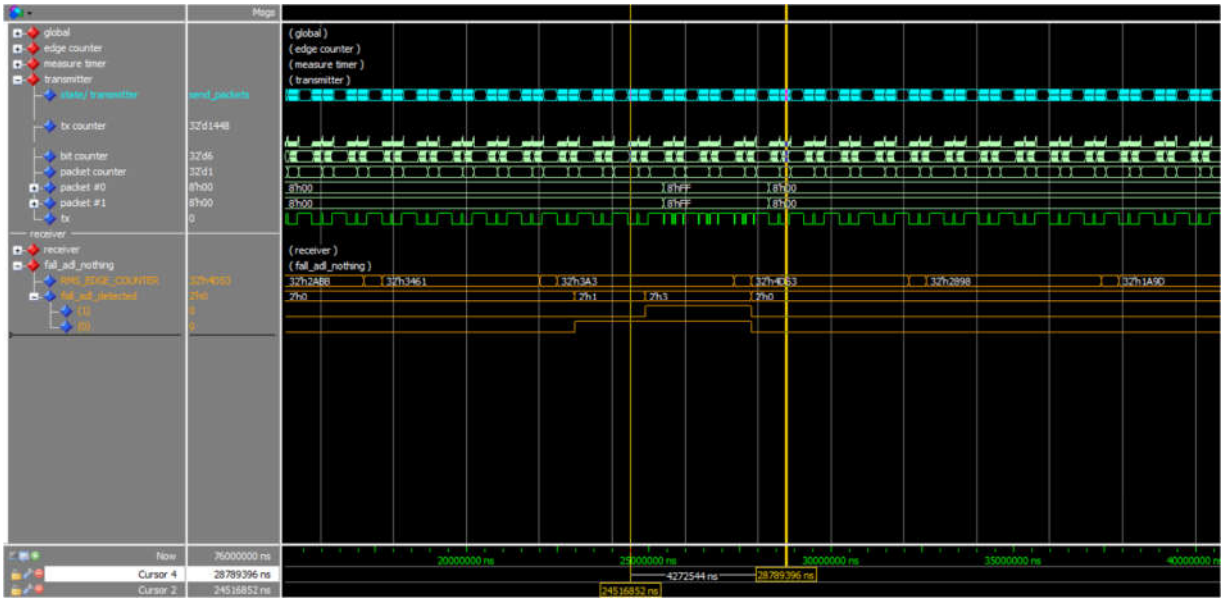
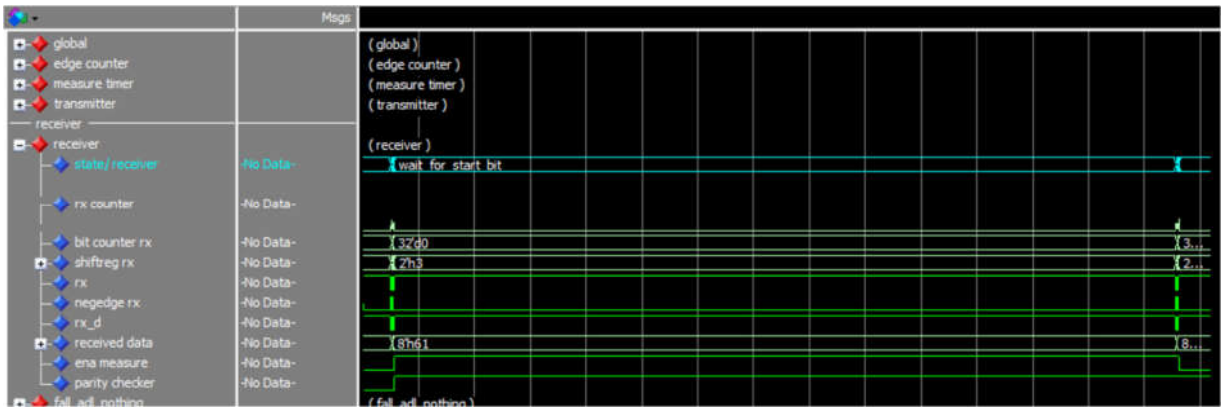


Figure 30 x"ffff" and X"0000" figure

17.2.3 7.3.2.1 UART Receiver verification

To test the receiver, we used X”61” (start bit) and X”62” (stop bit), which were transmitted to the receiver. Based on the received bit, only then the three counters can be seen to start working and stopping. When the bit was X”61” we can see the measurement process started, while at X”62” it stopped. As even parity was implemented in the transmission line data stream, it was also included in the receiver side to receive the transmitted data stream. The data received must match with the received parity bit, only then the data is fully validated, and the counters start working. Otherwise, the counters do not work [FIGURE].



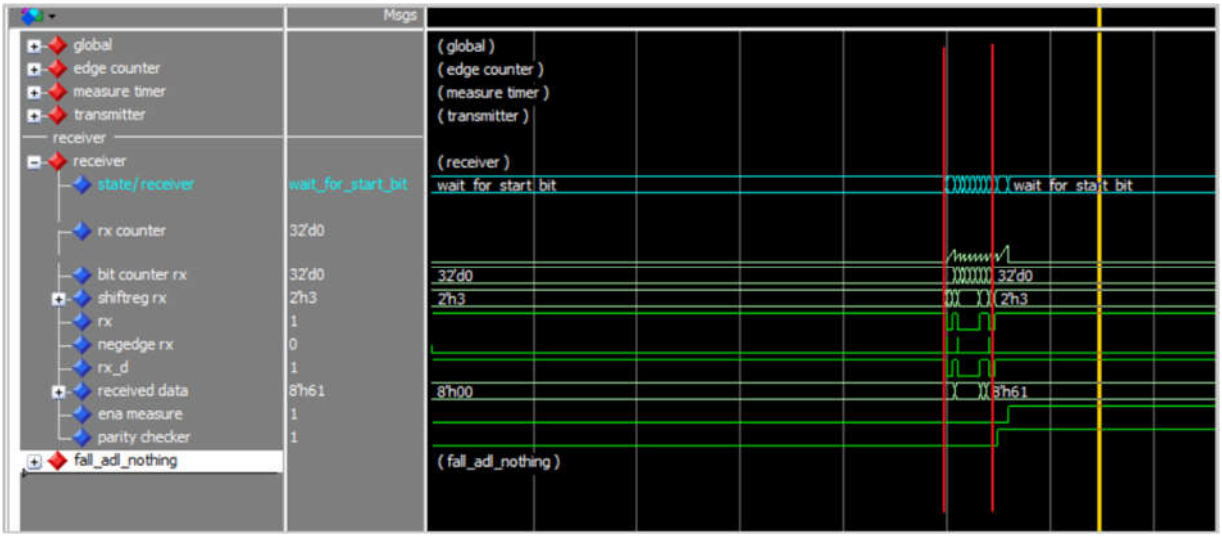


Figure 31parity wrong

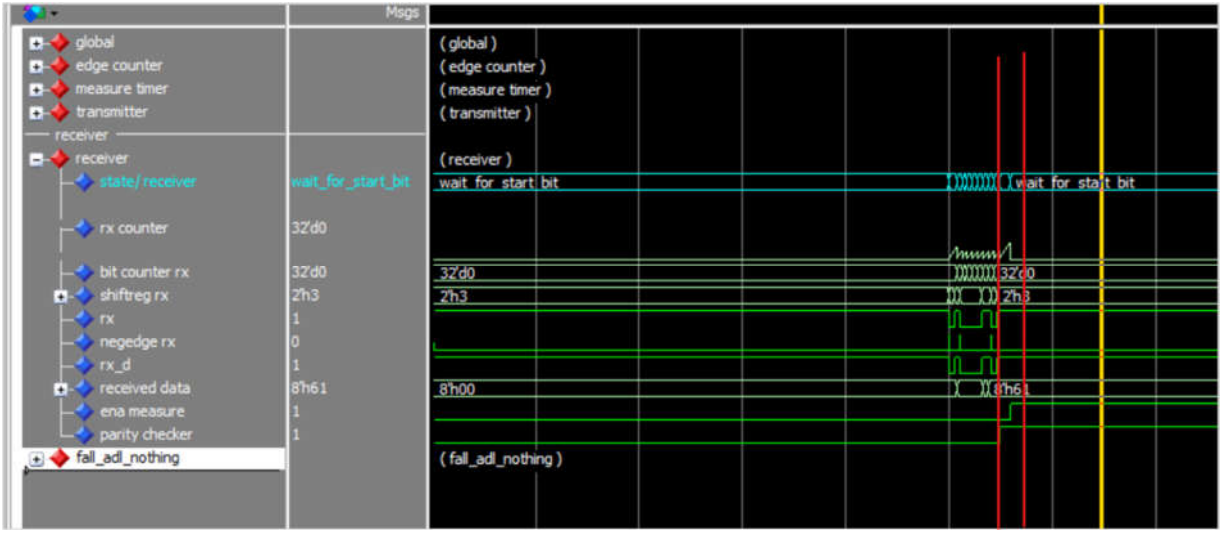


Figure 32 parity right

7.7. RTL synthesis on VHDL model (Quartus II)

For synthesizing of the VHDL design into an RTL model Quartus II was used, and the hardware FPGA Cyclone III (EP3C16F484) was selected as the modelling device. Alongside the VHDL code, another file called Synopsis Design Constraint file (extension .sdc) was passed to Quartus, in order to validate the pre-defined timing requirements by performing a time driven synthesis [FIGURE]. The aim of the file is to validate our design timing based on the minimum resource usage and the maximum allowable delays.

```
1   create_clock -period 20.0 -name clk [get_ports clk]
2
3   create_clock -period 200 -name freq_in  [get_ports freq_in]
4   create_clock -period 200 -name freq_in_2    [get_ports freq_in_2]
5   create_clock -period 200 -name freq_in_3    [get_ports freq_in_3]
6
7   create_clock -period 20.0 -name clk_ext
8
9   set_input_delay -clock clk_ext 0.2 [all_inputs]
10  set_output_delay -clock clk_ext 0.2 [all_outputs]
11  set_clock_groups -asynchronous -group {clk clk_ext} -group {freq_in freq_in_2 freq_in_3}
12  derive_clock_uncertainty
13
14
```

After the project was started with the selected VHDL code and the .sdc file (stored in the project directory), the analysis and synthesis flow was performed. [FIGURE] shows the successful VHDL synthesis in Quartus. As we can see from the summary it contains the fundamental information related to our used resources, combinational blocks and logical registers.

Flow Summary	
Flow Status	Successful - Sat Nov 28 20:52:55 2020
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	FMEAS_UART_ANZEN
Top-level Entity Name	FMEAS_UART_ANZEN
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	825
Total combinational functions	809
Dedicated logic registers	209
Total registers	209
Total pins	7
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	6
Total PLLs	0

7.8. RTL schematic of the circuit

The RTL schematic of our device is shown in Figure [FIGURE] which was accessed via the RTL viewer of the Netlist Viewer Tab in Quartus II. The schematic has also been segmented in a smaller part to zoom the contents ([FIGURE]) since it is too big. This is due to because of the number of combinational functions and logic registers being required by our design. The design shows the usage of these combinatorial and sequential networks via different interconnections.

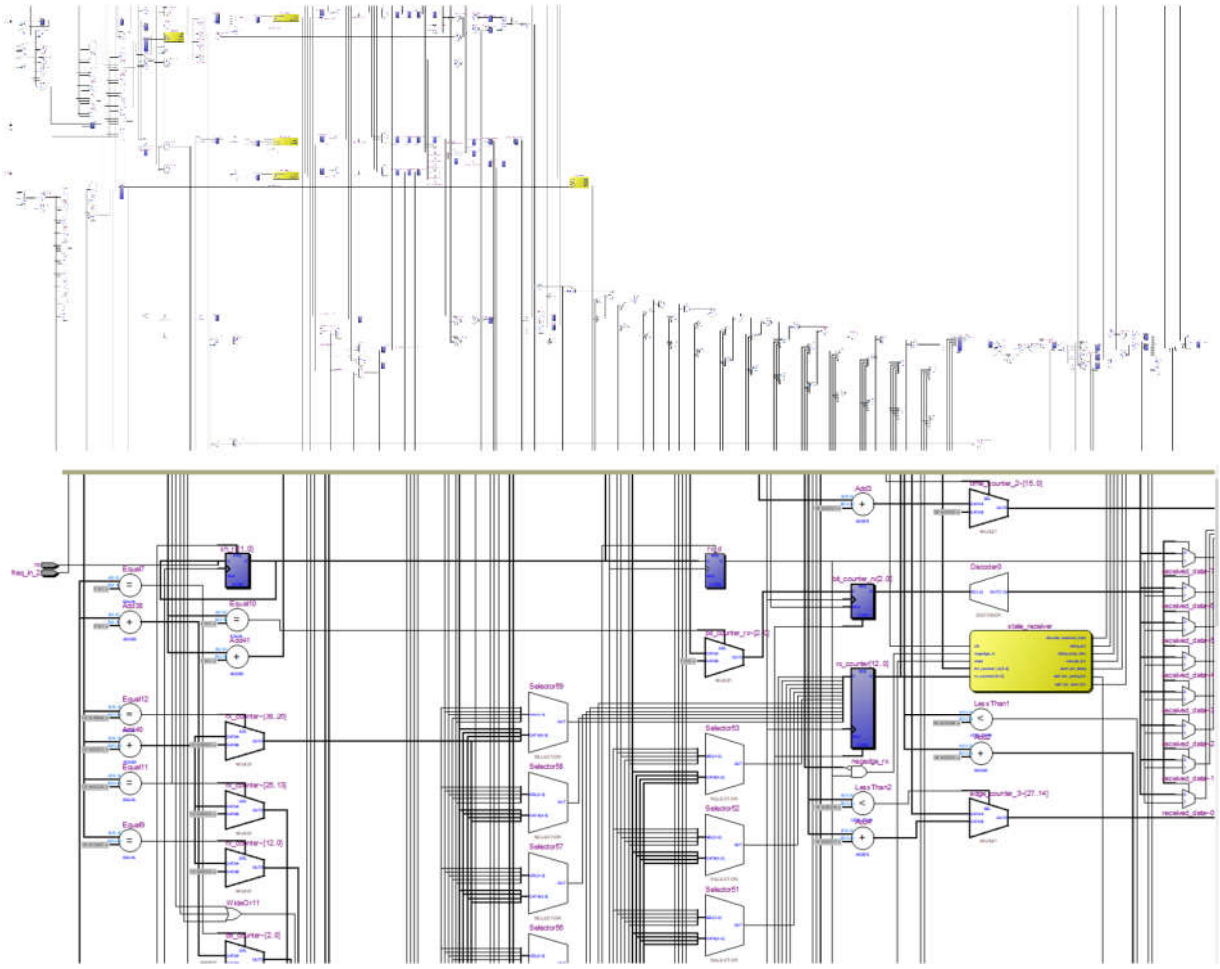
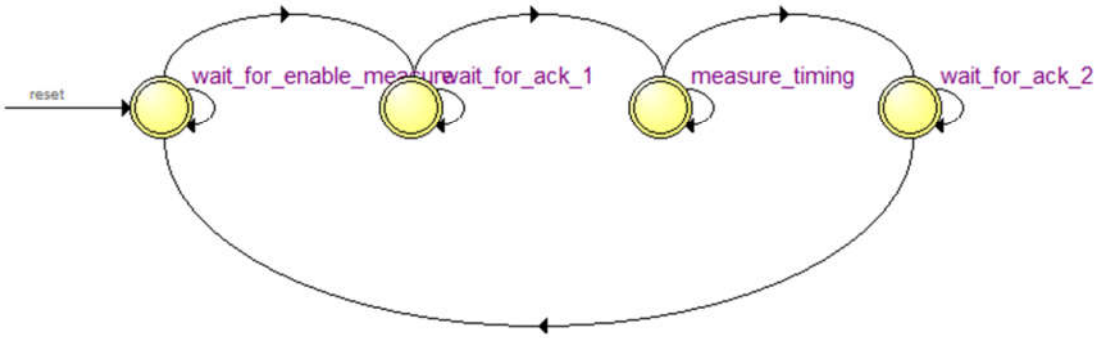
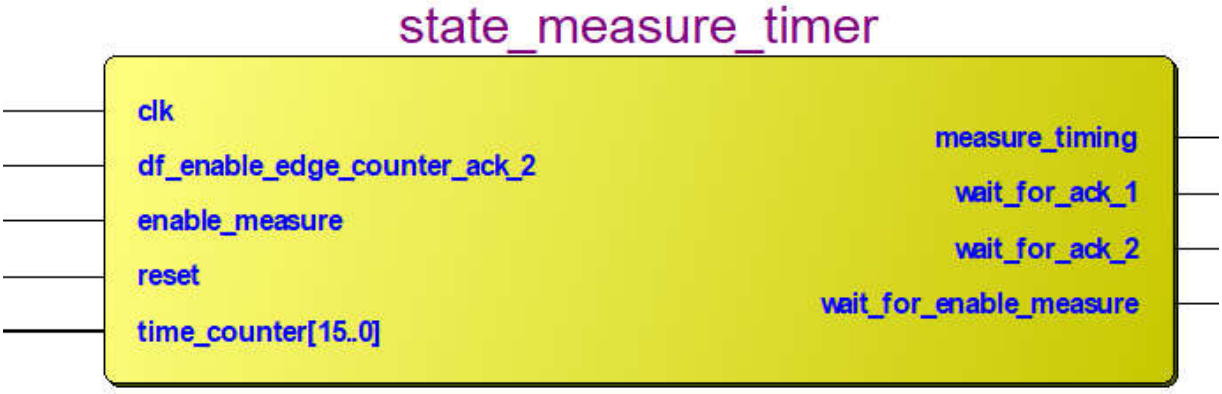
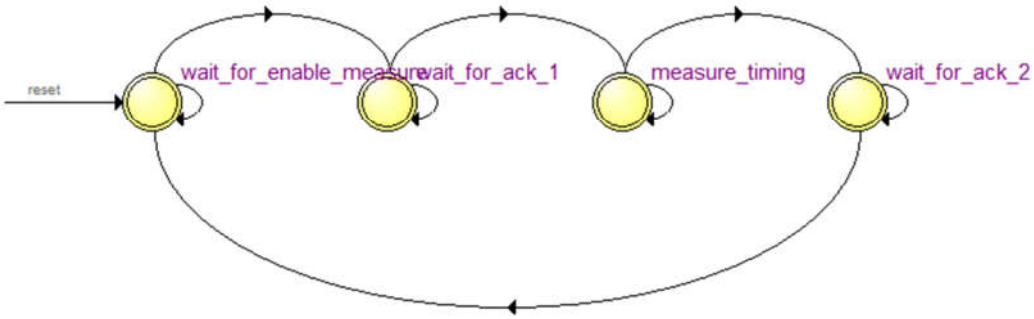


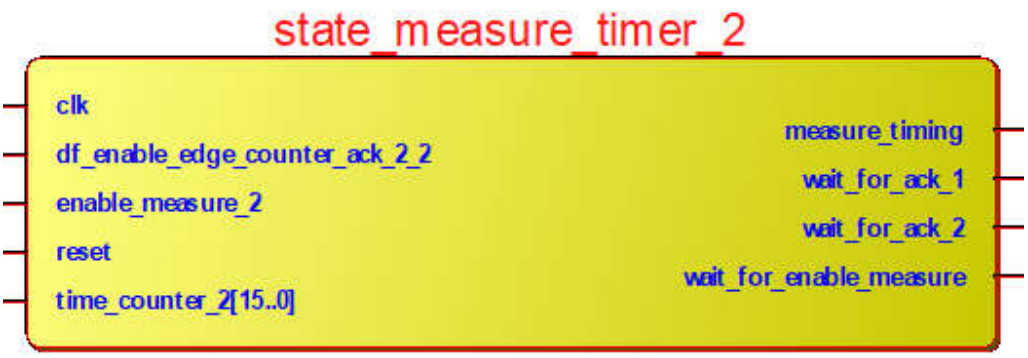
Figure 33 A section of the schematic (because it is too big)

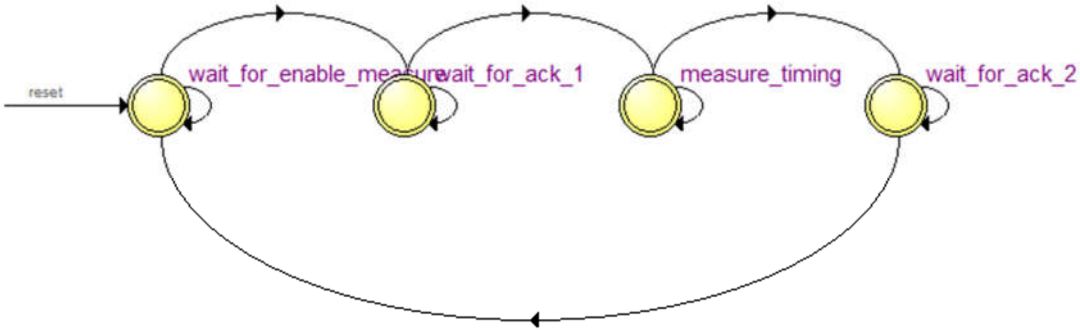
The state machines of our designed system were observed using the State Machine Viewer and RTL viewer tool under the same tab. [Figure] to [Figure] shows the generated state machines and their corresponding RTL block from Quartus II. We can see there are three separate state machines and RTL blocks for the three-frequency timer measurement. Apparently the three state machines have identical states, however their RTL model has different I/O ports.

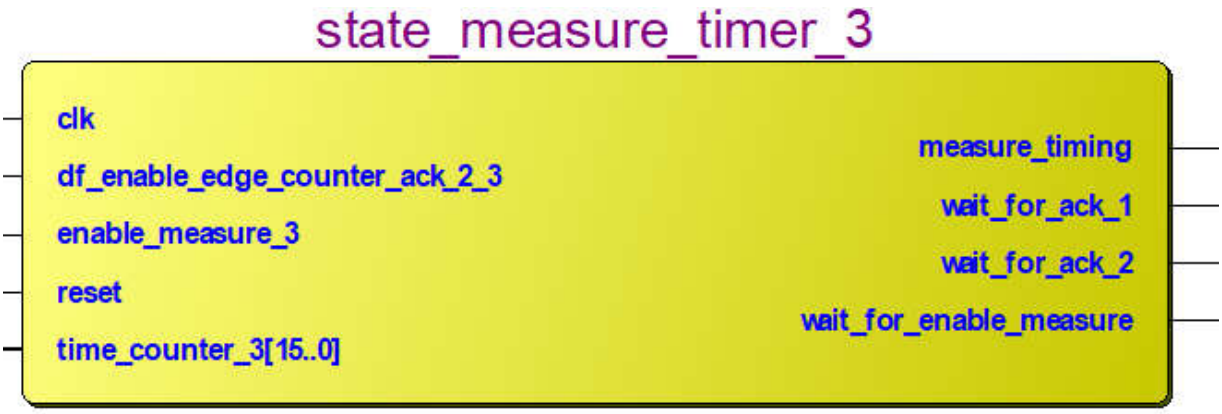

reset
wait_for_enable_measure
wait_for_ack_1
measure_timing
wait_for_ack_2

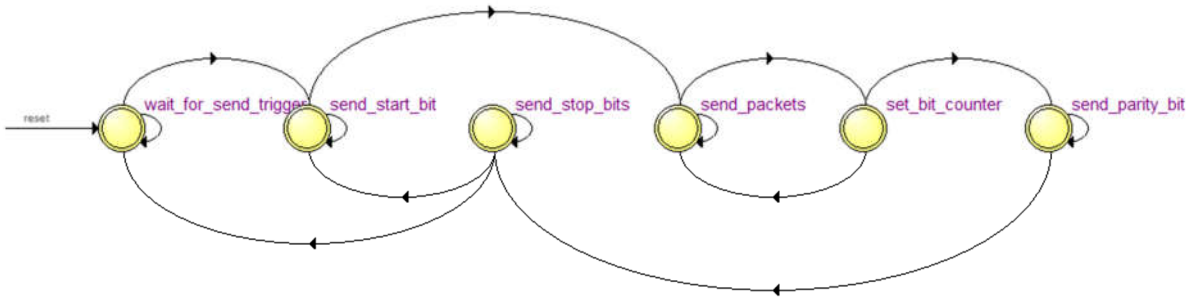

state_measure_timer
clk
df_enable_edge_counter_ack_2
enable_measure
reset
time_counter[15..0]
measure_timing
wait_for_ack_1
wait_for_ack_2
wait_for_enable_measure

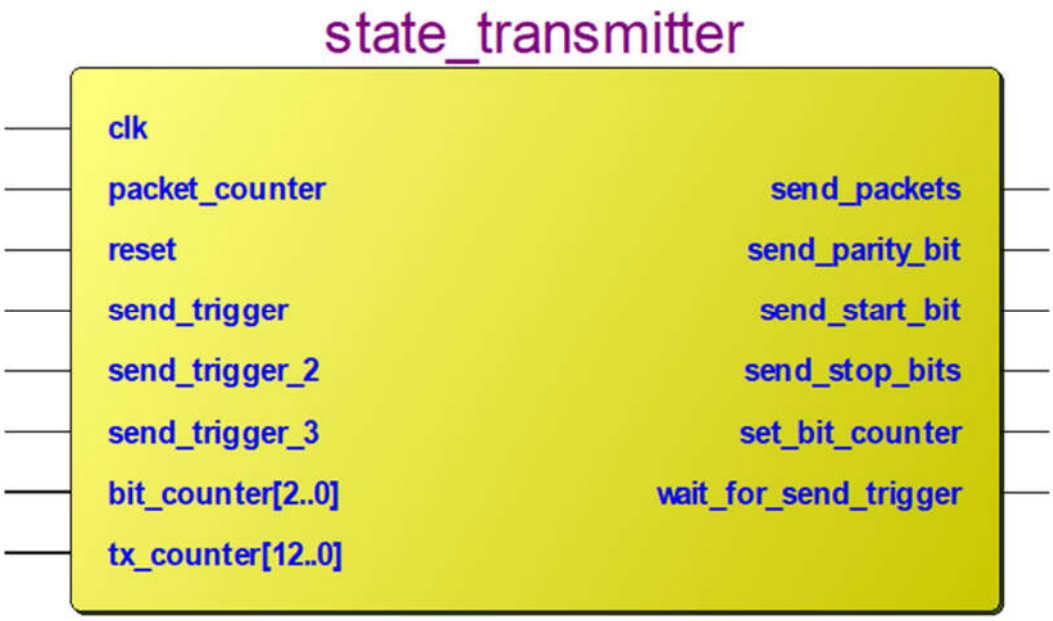

reset
wait_for_enable_measure
wait_for_ack_1
measure_timing
wait_for_ack_2

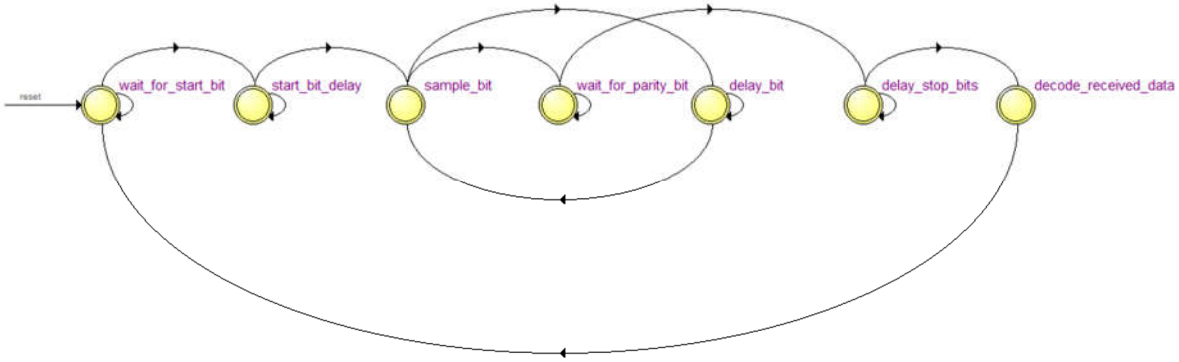

state_measure_timer_2
clk
df_enable_edge_counter_ack_2_2
enable_measure_2
reset
time_counter_2[15..0]
measure_timing
wait_for_ack_1
wait_for_ack_2
wait_for_enable_measure

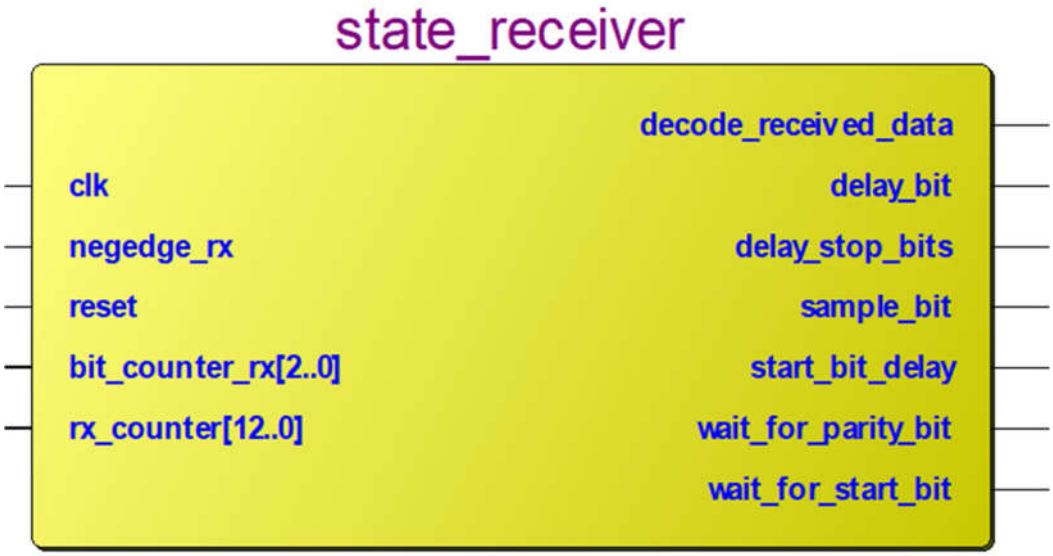

reset
wait_for_enable_measure
wait_for_ack_1
measure_timing
wait_for_ack_2


state_measure_timer_3
clk
df_enable_edge_counter_ack_2_3
enable_measure_3
reset
time_counter_3[15..0]
measure_timing
wait_for_ack_1
wait_for_ack_2
wait_for_enable_measure


reset
wait_for_send_trigger
send_start_bit
send_stop_bits
send_packets
set_bit_counter
send_parity_bit


state_transmitter
clk
packet_counter
reset
send_trigger
send_trigger_2
send_trigger_3
bit_counter[2..0]
tx_counter[12..0]
send_packets
send_parity_bit
send_start_bit
send_stop_bits
set_bit_counter
wait_for_send_trigger


reset
wait_for_start_bit
start_bit_delay
sample_bit
wait_for_parity_bit
delay_bit
delay_stop_bits
decode_received_data


state_receiver
clk
negedge_rx
reset
bit_counter_rx[2..0]
rx_counter[12..0]
decode_received_data
delay_bit
delay_stop_bits
sample_bit
start_bit_delay
wait_for_parity_bit
wait_for_start_bit

7.9. Pre-Layout Timing Analysis

The pre-layout timing analysis (or Static Timing Analysis (STA)) validates the design to meet the minimum timing requirements. This step is performed based on the SDC file mentioned earlier, which defines the timing constraints for the maximum allowable delays. STA compares these allowable maximum delays with the actual ones produced by the synthesis tool. However, since the constraint file can also be incorrect, another post-layout timing analysis was performed after the post-place & route simulation (discussed later).

The main purpose of the provided constraint file was to check whether our design can successfully operate on a FPGA with a global clock of 50 Mhz. We considered the results for the “Slow 1200mV 85C model” of the chosen FPGA in TimeQuest Timing Analyzer, because it contains the harshest environment and worst-case situation, including the temperature 85C.

We can see from the Fmax summary in Figure [FIGURE] that the maximum frequency, which can be reached by the system is 310.17 MHz. This value 10 times larger than our threshold frequency for fall detection, thus satisfying Nyquist-Shannon sampling theorem, as well as giving us large region to operate without errors.

Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name
1	226.86 MHz	226.86 MHz	freq_in_3
2	261.51 MHz	250.0 MHz	freq_in_2
3	277.78 MHz	250.0 MHz	freq_in
4	310.17 MHz	250.0 MHz	clk

The slack times of different clocks used in the circuit during the setup and hold time are given in Figure [FIGURE] and [FIGURE]. We can see that the slack times are all positive, that means that there is scope to increase the signal arrival time at different nodes without considerable affect on system delays. Also, a detailed slack of setup of the clock ‘clk’ is given in figure [FIGURE]. Like the previous observation, we can see that there are no negative slacks here. Thus, we can claim that our design does not any further adjustments, because the paths generated are quite fast to operate the circuit at the desired speed.

Slow 1200mV 85C Model Setup Summary

	Clock	Slack	End Point TNS
1	clk_ext	13.954	0.000
2	clk	16.776	0.000
3	freq_in_3	195.592	0.000
4	freq_in_2	196.176	0.000
5	freq_in	196.400	0.000

Slow 1200mV 85C Model Hold Summary

	Clock	Slack	End Point TNS
1	clk	0.358	0.000
2	freq_in	0.359	0.000
3	freq_in_2	0.359	0.000
4	freq_in_3	0.359	0.000
5	clk_ext	5.691	0.000

Slow 1200mV 85C Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	16.776	time_counter_2[9]	counter_ended_2	clk	clk	20.000	-0.080	3.139
2	16.779	time_counter_2[11]	counter_ended_2	clk	clk	20.000	-0.080	3.136
3	16.802	time_counter_2[9]	enable_edge_counter_2	clk	clk	20.000	-0.063	3.130
4	16.805	time_counter_2[11]	enable_edge_counter_2	clk	clk	20.000	-0.063	3.127
5	16.805	time_counter_2[15]	counter_ended_2	clk	clk	20.000	-0.080	3.110
6	16.818	time_counter_2[13]	counter_ended_2	clk	clk	20.000	-0.080	3.097
7	16.825	time_counter_2[3]	counter_ended_2	clk	clk	20.000	-0.081	3.089
8	16.831	time_counter_2[15]	enable_edge_counter_2	clk	clk	20.000	-0.063	3.101
9	16.844	rx_counter[3]	rx_counter[12]	clk	clk	20.000	-0.060	3.091
10	16.844	time_counter_2[13]	enable_edge_counter_2	clk	clk	20.000	-0.063	3.088
11	16.851	time_counter_2[3]	enable_edge_counter_2	clk	clk	20.000	-0.064	3.080
12	16.864	time_counter_2[8]	counter_ended_2	clk	clk	20.000	-0.081	3.050
13	16.890	time_counter_2[8]	enable_edge_counter_2	clk	clk	20.000	-0.064	3.041
14	16.917	time_counter[0]	time_counter[15]	clk	clk	20.000	-0.062	3.016
15	16.926	rx_counter[4]	rx_counter[11]	clk	clk	20.000	-0.062	3.007
16	16.928	rx_counter[1]	rx_counter[12]	clk	clk	20.000	-0.060	3.007
17	16.929	rx_counter[4]	rx_counter[5]	clk	clk	20.000	-0.062	3.004
18	16.930	rx_counter[4]	rx_counter[1]	clk	clk	20.000	-0.062	3.003
19	16.946	time_counter[12]	enable_edge_counter	clk	clk	20.000	-0.057	2.992
20	16.951	time_counter[12]	counter_ended	clk	clk	20.000	-0.057	2.987
21	16.951	time_counter_2[14]	counter_ended_2	clk	clk	20.000	-0.080	2.964
22	16.958	time_counter[1]	enable_edge_counter	clk	clk	20.000	-0.057	2.980
23	16.959	time_counter_2[7]	counter_ended_2	clk	clk	20.000	-0.081	2.955
24	16.963	time_counter[1]	counter_ended	clk	clk	20.000	-0.057	2.975
25	16.967	rx_counter[12]	rx_counter[4]	clk	clk	20.000	-0.061	2.967
26	16.968	time_counter_2[0]	counter_ended_2	clk	clk	20.000	-0.080	2.947

7.10. Resource requirements and the timing characteristics of synthesized circuit

After the STA, the Fitter (Place & Router) analysis was performed. Through this operation, the cells of mapped logic gate level models are fitted to exact locations, and the routing resources are dynamically allocated to implement the networks. This helped us to gather necessary details about the resource requirements and timing characteristics of the synthesized circuit. [FIGURE] shows the summary of the Fitter flow, while [FIGURE] and [FIGURE] shows the detailed one. We can see that the design is using about 6% of the provided resources, which is much low.

Flow Summary	
Flow Status	Successful - Sat Nov 28 22:10:03 2020
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	FMEAS_UART_ANZEN
Top-level Entity Name	FMEAS_UART_ANZEN
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	866 / 15,408 (6 %)
Total combinational functions	851 / 15,408 (6 %)
Dedicated logic registers	209 / 15,408 (1 %)
Total registers	210
Total pins	7 / 347 (2 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	6 / 112 (5 %)
Total PLLs	0 / 4 (0 %)

Fitter Resource Usage Summary

	Resource	Usage
1	⌄ Total logic elements	866 / 15,408 (6 %)
1	-- Combinational with no register	657
2	-- Register only	15
3	-- Combinational with a register	194
2		
3	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	138
2	-- 3 input functions	347
3	-- <=2 input functions	366
4	-- Register only	15
4		
5	⌄ Logic elements by mode	
1	-- normal mode	422
2	-- arithmetic mode	429
6		
7	⌄ Total registers*	210 / 17,068 (1 %)
1	-- Dedicated logic registers	209 / 15,408 (1 %)
2	-- I/O registers	1 / 1,660 (< 1 %)
8		
9	Total LABs: partially or completely used	66 / 963 (7 %)
10	Virtual pins	0
11	⌄ I/O pins	7 / 347 (2 %)
1	-- Clock pins	3 / 8 (38 %)
2	-- Dedicated input pins	0 / 9 (0 %)
12		

12		
13	Global signals	5
14	M9Ks	0 / 56 (0 %)
15	Total block memory bits	0 / 516,096 (0 %)
16	Total block memory implementation bits	0 / 516,096 (0 %)
17	Embedded Multiplier 9-bit elements	6 / 112 (5 %)
18	PLLs	0 / 4 (0 %)
19	Global clocks	5 / 20 (25 %)
20	JTAGs	0 / 1 (0 %)
21	CRC blocks	0 / 1 (0 %)
22	ASMI blocks	0 / 1 (0 %)
23	Impedance control blocks	0 / 4 (0 %)
24	Average interconnect usage (total/H/V)	0% / 0% / 0%
25	Peak interconnect usage (total/H/V)	3% / 4% / 3%
26	Maximum fan-out	209
27	Highest non-global fan-out	38
28	Total fan-out	3098
29	Average fan-out	2.80

7.11. Post-place & route simulation model and timing Analysis

To have a possibility to import the post-place&route simulation model it was necessary to create it from level of Quartus II. This option is given by EDA Netlist Writer. After that the simulation files: FMEAS_UART_6_1200mv_85c_slow.vho and FMEAS_UART_6_1200mv_85c_vhd_slow.sdo appeared in the simulation folder. Through changing names of the entities both designs were compared: previously prepared one and the one generated through Quartus II.

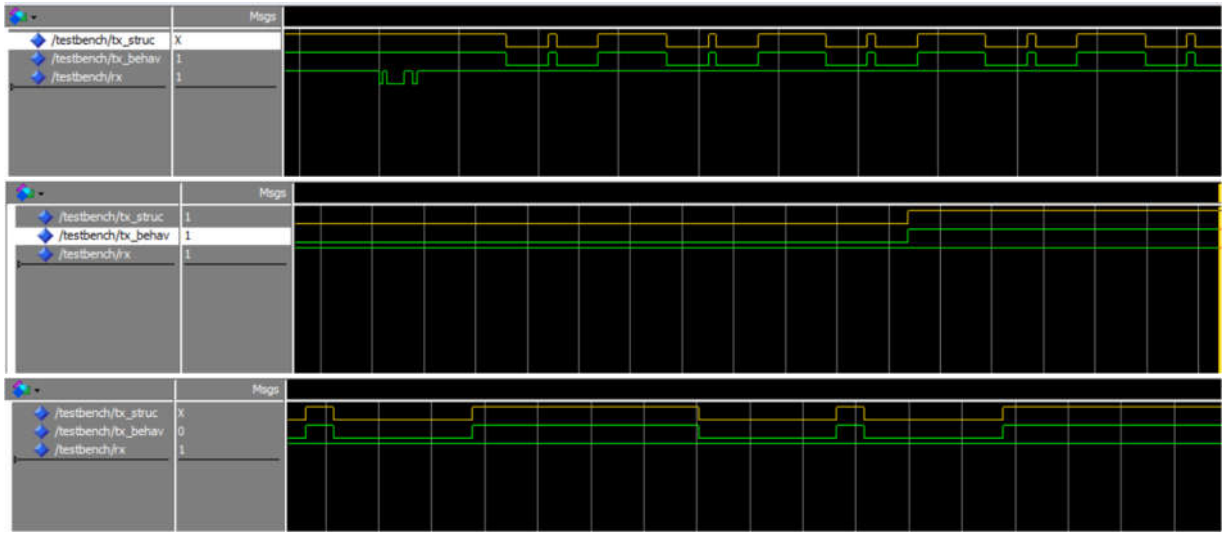
To detect possible differences between designs following signals were chosen: rx (from testbench), tx_struct (L_DUT_STRUC, from Quartus II design), tx_behav (L_DUT_BEHAV from previously prepared design). The simulation had to be run for 25ms (which in real time took about one hour).

The post-place and route simulation was performed by writing the EDA netlist. This was performed using the EDA Netlist Writer. This was done using the EDA Netlist Writer. A set of simulation files were generated, and we chose the FMEAS_UART_ANZEN_6_1200mv_85c_slow.vho and FMEAS_UART_ANZEN_6_1200mv_85c_vhd_slow.sdo files. As we analyzed using the 85C slow model in the pre-layout timing analysis stage, we are selecting the same model files. The vho file corresponds to the RTL model of our digital system, while the sdo file contained the delay data of the technology cells and their interconnections of the selected post-place and route simulation model. We also changed the names of the structural entity and the behavioral entities for the timing analysis.
The following signals were chosen to analyze the possible differences in timing and delay between the pre-layout and post-layout design:

- rx
- tx_struct (from the Quartus II RTL model)

- tx_behav (from the previous VHDL design)

SDF file was included during the simulation performed in ModelSim Altera. The simulation was run for 25 ms.



The simulated waveforms from the updated testbench is given in Figure [FIGURE] to [FIGURE]. We can see that no delays were noted, even in the zoomed images. This verifies our digital system's timing requirement and will ensure proper sensor operations.

References

[1] A. Stapleton, "Efficacy of Methylphenidate in the Geriatric Population for Fall Prevention," 2018.

[2] N. El-Bendary, Q. Tan, F. C. Pivot, and A. Lam, "FALL DETECTION AND PREVENTION FOR THE ELDERLY: A REVIEW OF TRENDS AND CHALLENGES," *International Journal on Smart Sensing & Intelligent Systems,* vol. 6, no. 3, 2013.

[3] W. H. O. Who, "Falls," (in English), *World Health Organization: WHO,* 2018/01/16/ 2018. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/falls.

[4] N. Adamczewska and S. R. Nyman, "A New Approach to Fear of Falls From Connections With the Posttraumatic Stress Disorder Literature," (in eng), *Gerontol Geriatr Med,* vol. 4, pp. 2333721418796238-2333721418796238, 2018, doi: 10.1177/2333721418796238.

[5] H. Gjoreski, M. Lustrek, and M. Gams, "Accelerometer placement for posture recognition and fall detection," in *2011 Seventh International Conference on Intelligent Environments*, 2011: IEEE, pp. 47-54.

[6] P. Ntanasis, E. Pippa, A. T. Özdemir, B. Barshan, and V. Megalooikonomou, "Investigation of sensor placement for accurate fall detection," in *International Conference on Wireless Mobile Communication and Healthcare*, 2016: Springer, pp. 225-232.

[7] A. T. Özdemir, "An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice," *Sensors,* vol. 16, no. 8, p. 1161, 2016.

[8] C.-N. Huang, C.-Y. Chiang, G.-C. Chen, S. Hsu, W.-C. Chu, and C.-T. Chan, "Fall detection system for healthcare quality improvement in residential care facilities," *J. Med. Biol. Eng,* vol. 30, no. 4, pp. 247-252, 2010.

[9] "Tango® Belt - Hip Protection Redefined," ed, 2020.
[10] "Hip'Safe by Helite – Airbag protection for seniors," ed, 2020.
[11] X. Wang, J. Ellul, and G. Azzopardi, "Elderly Fall Detection Systems: A Literature Survey," *Frontiers in Robotics and AI,* vol. 7, 2020, doi: 10.3389/frobt.2020.00071.
[12] A. Kurniawan, A. R. Hermawan, and I. K. E. Purnama, "A wearable device for fall detection elderly people using tri dimensional accelerometer," 2016: IEEE, doi: 10.1109/isitia.2016.7828740. [Online]. Available: https://dx.doi.org/10.1109/isitia.2016.7828740
[13] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "SisFall: A Fall and Movement Dataset," (in eng), *Sensors (Basel),* vol. 17, no. 1, p. 198, 2017, doi: 10.3390/s17010198.
[14] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "Real-Life/Real-Time Elderly Fall Detection with a Triaxial Accelerometer," (in eng), *Sensors (Basel),* vol. 18, no. 4, p. 1101, 2018, doi: 10.3390/s18041101.
[15] P. Pierleoni, A. Belli, L. Palma, M. Pellegrini, L. Pernini, and S. Valenti, "A high reliability wearable device for elderly fall detection," *IEEE Sensors Journal,* vol. 15, no. 8, pp. 4544-4553, 2015.
[16] A. Purwar, D. U. Jeong, and W. Y. Chung, "Activity monitoring from real-time triaxial accelerometer data using sensor network," in *2007 International conference on control, automation and systems*, 2007: IEEE, pp. 2402-2406.
[17] "Google Trends," ed, 2020.
[18] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online,* vol. 12, no. 1, p. 66, 2013.
[19] A. K. Bourke, J. V. O'Brien, and G. M. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," (in eng), *Gait Posture,* vol. 26, no. 2, pp. 194-9, Jul 2007, doi: 10.1016/j.gaitpost.2006.09.012.
[20] M. Saleh and R. L. B. Jeannès, "Elderly Fall Detection Using Wearable Sensors: A Low Cost Highly Accurate Algorithm," *IEEE Sensors Journal,* vol. 19, no. 8, pp. 3156-3164, 2019, doi: 10.1109/JSEN.2019.2891128.
[21] K. Chaccour, R. Darazi, A. H. el Hassans, and E. Andres, "Smart carpet using differential piezoresistive pressure sensors for elderly fall detection," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2015: IEEE, pp. 225-229.
[22] M. Pickavance, "Best fall detection sensors of 2020," (in English), *TechRadar,* 2020/03/31/ 2020. [Online]. Available: https://www.techradar.com/best/best-fall-detection-sensors.
[23] S. Timoshenko and J. N. Goodier, "Strength of Materials Part I," *J. Elast.*, 1955.
[24] Oktaviani.J, *済無No Title No Title*, vol. 51, no. 1. 2018.
[25] G. Brown, "An accelerometer based fall detector: development, experimentation, and analysis," *University of California, Berkeley*, no. July. pp. 1–9, 2005.
[26]E. H. Klaassen *et al.*, "Silicon fusion bonding and deep reactive ion etching: a new technology for microstructures," *Sensors and Actuators A: Physical,* vol. 52, no. 1-3, pp. 132-139, 1996.
[27] S. T. Cho, "Batch-dissolved wafer process for low-cost sensor applications," in *Micromachining and Microfabrication Process Technology*, 1995, vol. 2639: International Society for Optics and Photonics, pp. 10-17.
[28]J. Dong, Z.-j. Long, H. Jiang, and L. Sun, "Monolithic-integrated piezoresistive MEMS accelerometer pressure sensor with glass-silicon-glass sandwich structure," *Microsystem Technologies,* vol. 23, no. 5, pp. 1563-1574, 2017.

[29]F. Ender. B. U. o. technology. (2020). Redeuced order Modelling in Ansys.
[30]H.-I. Jung, D.-S. Kwon, and J. Kim, "Fabrication and characterization of monolithic piezoresistive high-g three-axis accelerometer," *Micro and Nano Systems Letters,* vol. 5, no. 1, p. 7, 2017.
[31]G. Tang *et al.*, "Fabrication and analysis of high-performance piezoelectric MEMS generators," *Journal of Micromechanics and Microengineering,* vol. 22, no. 6, p. 065017, 2012.
[32]S. Tez and T. Akin, "Comparison of two alternative fabrication processes for a three-axis capacitive MEMS accelerometer," *Procedia Engineering,* vol. 47, pp. 342-345, 2012.
[33]Y. Zhou, Z. Wang, Q. Zhang, W. Ruan, and L. Liu, "A Front-Side Released Single Crystalline Silicon Piezoresistive Microcantilever Sensor," *Sensors Journal, IEEE,* vol. 9, pp. 246-254, 04/01 2009, doi: 10.1109/JSEN.2008.2012197.
[34] C. Wang, J. Zeng, K. Zhao, and H. Chan, "Chip scale studies of BCB based polymer bonding for MEMS packaging," in *2008 58th Electronic Components and Technology Conference*, 2008: IEEE, pp. 1869-1873.