Project: VHDL-AMS & MEMS Accelerometer

Authors:

GROUP 12: Fabio Grassi, Stefano Iuliano, Alessandro Nadal.

Index

1.0 Introduction

Nowadays researching and designing are based on automatic and computerized tools. Paper is useful to reach a basic idea of the project allowing the designer to implement it on a computer. Micro and Nano System designing techniques need powerful tools because of complexity of MEMS and NEMS. Our aim is demonstrate how MatLab Simulink and the VHDL-AMS language can be a valid support for MEMS developing. In particular, our attention is on the designing and simulation of an accelerometer for car airbag application.

2.0 VHDL-AMS

2.1 Introduction on VHDL-AMS

Nowadays with the advent of MEMS and NEMS, we need tools that allow the interaction between different physicals domains. Examples are electrical, thermal, mechanical, optical and chemical etc. Everyone needs a set of physical variables and parameters to describe phenomena. If we study every system alone, neglecting the interfaces and cross coupling, we can fall in consistent errors and problems like loss in performances or wrong modelling. One way to take into account the interactions is to use a simulation tool that allows the description and simulation of systems from different domains. VHDL-AMS and Verilog-AMS are an example. In fact, thanks to these high-level description languages, we can design systems including both discrete time and continuous time models with multiple energy domains. Modeling is the most important thing inside any project. The description of the system and phenomena that we want analyze is important: it defines the quality of our results and the final realization. Model may describe the behavior at various levels of detail. Every choice among the possible levels compromise the accuracy and performance. Another important aspect is the model complexity. Usually, deep level of description, means higher complexity.
Time in every physical system is used for different purpose. In a discrete system time is the reference of every event while in continuous system is used for equations solving. Since it has a minimum resolution, it compromises simulation and results.
Another important aspect is the data abstraction level. Every signal or variable can be digital with Boolean value or analog with real value. The resolution chosen is important to get good results during the modeling.

Since the data abstraction level is usually related to the time abstraction we can delineate two different types of model:

- Discrete-Time model where we have logic functions of discrete variables;
- Continuous-Time model where we have real functions of continuous variables.

Two models have also different equations that describe their behavior. In Discrete-Time models we have logical Boolean equations and the

communication between different blocks happens with trigger events. In Continuous-Time models instead, we have differential algebraic equations. The basic idea for any design process is to divide a system in a set of simpler parts in order to create a hierarchy that allow a simpler designing, simulation and debugging. The interconnection between different blocks is defined by elements called ports. Their characteristics are the directional of signal flow (for both discrete and continuous time models) and the conservative-law relationship between quantities (for continuous time models only). The conservative-law relationship is based on two different quantities namely across quantities and through quantities. The first one represents an effort (for example voltage for electrical system, position for mechanical system). The second one represents a flow (for example current for electrical system, force for mechanical system).

2.2 Characteristics

The VHDL-AMS is based on VHDL (IEEE 1076-1993) and it is described by IEEE 1076.1-1999. The acronym AMS corresponds to Analog-Mixed-Signal and it is an extension of the standard VHDL. It is able to represent complex models directly using both Non-Linear Ordinary Differential Algebraic Equations and digital signals. VHDL-AMS has a concept of time based on concurrent processes. A generic model is organized in two different parts: Entity and Architecture.

The entity is used to describe the ports of system and their nature. It is also used to define constant values used as a parameters inside the architecture description. Our example is based on a resistor. We can describe its Entity as:

```
ENTITY Resistor IS

GENERIC (R0: REAL:= 100.0);

PORT (
TERMINAL p1, p2: electrical);

END ENTITY Resistor;
```

As we can see the names “Resistor”, “p1” and “p2” are totally user defined. The yellow parts are imposed by the VHDL-AMS syntax. The word “REAL” describe the type of parameter R0. Its value is 100. Finally the name “electrical” is the

nature of the two terminals.

The architecture instead is used to describe what happens through and across the two terminals when is applied an external stimulus. In this case we have to describe the relation between voltage and current. An example of the architecture is:

```
ARCHITECTURE ideal OF Resistor IS

QUANTITY V ACROSS I THROUGH p1 TO p2;

BEGIN

I== V/ R0;

END ARCHICTECTURE Resistor;
```

As we can see we have two parts. Before the BEGIN statement we have first one that is used to define constant parameters or quantity variables, in our case voltage and current. The second one, after the BEGIN statement, is used to define the equations of our model, in this case the Ohm's law. Every equation or digital process is contemporary with respect the other, so no matter which is the order of the equations. Every statement is terminated by semicolon character. In the first code example, we can modify the value of the resistor because in a traditional VHDL simulation we declare a component using the following statement:

```
Resistor1:  Resistor GENERIC MAP (10) PORT MAP (P1, P2);
```

"Resistor1" is the label that identify the instance Resistor. Through the GENERIC MAP we assign every parameter and terminal in the same order described inside the entity. If we do not define GENERIC MAP the simulation assumes the default value declared inside the entity. The PORT MAP allows to connect different components. In this way, the series of two Resistor becomes:

```
Resistor1:  Resistor GENERIC MAP (10) PORT MAP (P1, P2);

Resistor2:  Resistor GENERIC MAP (33) PORT MAP (P2, P3);
```

Another possible implementation is to define the value of resistor as a constant inside the architecture removing the generic statement inside the entity.

```
ENTITY Resistor IS

PORT (
TERMINAL p1, p2: electrical);

END ENTITY Resistor;

ARCHITECTURE ideal OF Resistor IS

QUANTITY R0: REAL: = 100.0;

QUANTITY V ACROSS I THROUGH p1 TO p2;

BEGIN

I== V/ R0;

END ARCHICTECTURE ideal;
```

In this example we cannot modify individually the value of two resistors because they are defined once inside VHDL-AMS model.

2.2.1 VHDL-AMS Object Types

There are six classes of objects in VHDL-AMS:

- *Constants*;
- *Terminals*;
- *Quantities*;
- *Generic Constants*;
- *Digital Ports*;
- *Signals*;

Constants

Constants are object used to storage values used inside the model. Examples of constants declaration are:

- CONSTANT R0: REAL:= 10.0;
- CONSTANT n: INTEGER:= 10;

The main purpose of constants declaration is to simplify the changing of one or more values inside the model. Imagining to have many equations that use a parameter, if we did not define it as a constant and we want to update it we must change it in every point of the model manually. Another important

benefit is the readability of the code. Constants cannot change during the simulation time.

Terminals

They represent the continuous and conservative ports and have an across effort and a through flow. Every terminal has own nature that depends on the physical domain. Examples of predefined terminals are:

- *Electrical:* Voltage is Across and Current is Through;
- *Translational:* Position is Across and Force is Through;
- *Thermal:* Temperature is Across and Entropy Flow Rate is Through;
- *Fluidic:* Pressure is Across and Volumetric Flow Rate is Through;

It is also possible to define custom terminal natures.

Quantities

There are three types of quantities:

- *Free Quantities*: they are non-conservative analog objects used to clarify the model description and they can change during simulation:
  - QUANTITY A: REAL;
- *Branch Quantities:* they are analog objects used for conservative energy systems and they are used with terminals declared inside the entity:
  - QUANTITY V ACROSS I THROUGH P1 TO P2;
- *Source Quantities:* they are used for frequency domain modeling:
  - QUANTITY bode REAL SPECTRUM magnitude, phase;

Generic Constants

As we have seen previously the generic statements allow models to be externally parameterized. Generic constants are define inside the entity of our model and allow to change a parameter inside, without changing the model itself. They are also useful to define different instances of our model with different constant parameters.

Digital Ports

They represent the discrete ports for the digital world. Every port has a directional statement that defines its type. Examples of predefined digital ports are:

- *IN* for Input ports;
- *OUT* for Output ports;
- *INOUT* for Bidirectional ports;
- *BUFFER* for Output ports readable internally;

Every digital port can bring a specific type of digital signal (bit, std_logic, unsigned, etc.). Examples of digital port declaration is:

```
PORT (
A, B: IN bit;
Y: OUT bit;
C: IN unsigned (3 downto 0));
```

There are many types of digital input/output ports and they are described inside IEEE packages.

Signals

Signals represent digital variables inside our system. They are declared inside the architecture and are used as free quantities in continuous domains. A signal has own name and own type. If we want to connect an internal signal to an external digital port we have to use the same type. Examples of signals declaration are:

```
SIGNAL A, B: bit:='0';
SIGNAL Y: bit;
SIGNAL C: unsigned ( 3 downto 0);
```

Signals have not statements that specify if they are input or output. They can be connected directly on the desired digital port. As we can see there is the possibility to define also ports and signals with a parallelism higher than 1 bit with the statement "(n downto 0)" or "(0 to n)". The statement depends on the logic justification used inside the system.

Other Consideration

Inside VHDL-AMS it is possible to use also user define domains and objects. Everyone is described inside a package that is a particular VHDL file that allows to define own libraries, components and objects. For every signals and quantities we can also define vector structures to simplify the organization of variables inside the model.

2.2.2 Other: Continuous Models

Implicit Quantities

Assuming a quantity called X, we can define these associated quantities:

- X'Dot: the derivative of quantity X with respect the time;
- X'Integ: the integral of quantity X over time from zero to current time;
- X'Delayed(T): quantity X delayed by T (ideal delay, T must be >=0);
- X'Ltf(num,den): Laplace transfer function whose input is X;
- Q'Ztf(num, den, T, initial_delay): Z-Domain transfer function whose input is X.

If-Elsif-Else Statement

Inside the Architecture, after the begin statement, we can use If-Elsif-Else statement to check a quantity in order to define the value of another one. An example is proposed:

```
IF Input > Threshold USE

VOut == 1.0;

ELSIF Input < (-Threshold) USE

VOut == -1.0;

ELSE VOut == 0.0;

END USE;
```

2.2.3 Other: Discrete Models

If-Elsif-Else Statement

Inside an Architecture or a Process, we use the If-Elsif-Else statement to check the state of one or more signals (through logical expressions) in order to define the value of another set of signals. An example is proposed:

```
IF (Input = '0') THEN
Output <= '0';
ELSIF (Input = '1' AND Flag='0') THEN
Output <= '1';
ELSE
Output<=Output;
END IF;
```

Process Statement

Inside a digital system we could need to describe both combinatorial and sequential logic. Sequential logic has a clock and its outputs have a memory state. In order to describe this kind of logic, the VHDL introduces the Process Statement. An example is given:

```
LABEL: PROCESS (Sensitivity List)
BEGIN
Body of Process
END PROCESS;
```

The LABEL represents the name of our process and it is useful to create readable model description. Sensitivity list represents the set of input signals that can invoke the process. Finally the body of process is the set of Boolean expressions to generate the process outputs.

An example of process is the following D flip-flop.

```
FF: PROCESS (clock,Reset,D)
BEGIN
IF(clock'EVENT and CLOCK='1' AND Reset='1') THEN
Output<=D;
ELSIF(clock'EVENT and CLOCK='1' AND Reset='0')THEN
Output<='0';
END IF;
```

2.2.4 Analog Digital Interface

Since inside entity we can declare both digital and analog port, we have to interface their inside the architecture. The statement used is a particular process. An example of comparator with analog input and digital output is proposed.

```
PROCESS (Vin' ABOVE(Threshold)) IS
BEGIN
IF Vin'ABOVE(Threshold) THEN
OutComparator<='1';
ELSE
OutComparator <='0';
END IF;
END PROCESS;
```

As we can see the sensitivity list includes the analog event (Vin' ABOVE(Threshold)) that invokes the process. The syntax inside is the standard VHDL used for discrete time modelling.

2.3 Simulation in VHDL-AMS

The simulation through VHDL-AMS needs a powerful numerical solver in order to solve the Differential Algebraic Equations of our model. Usually the differential equations are discretized using a fixed or variable time step calculating an approximated solution as a piecewise linear functions of time. The quality of the solution depends on the tolerances. They are defined in relation with the discretization time step. These approximations are enhanced in mixed-signal simulation model because of the presence of digital system. Here we need an additional mechanisms to combine both discrete and continuous results and finally represent them. The digital simulation is event based and for this reason has a simpler computation.

3.0 Project

3.1 Introduction on Accelerometers

Accelerometers are useful sensors that can detect an acceleration along one axis or more like in 2D or 3D systems. The main types of accelerometers are:

- Capacitive Accelerometer: they are based on mass-spring-damper system and convert the displacement between two plates into a capacitance variation. The mass is realized with a conductive material and it corresponds to the movable plate. They can be based on a single capacitor or on a differential capacitor structure. First one is used to measure only positive acceleration while the second one is used for both positive and negative;

- Extensometer Accelerometer: they are based on a seismic mass connected to one or more extensometers. The position variation is translate into a resistance. They strongly depend on the temperature;

- Piezoelectric Accelerometer: they are based on a seismic mass mechanically connected to a piezoelectric crystal. It generates an electrical signal when a compression occurs. This type of accelerometer senses only dynamic acceleration and it cannot be used for static one. Piezoelectric accelerometers have high input range (hundreds of g) and they cannot be used for small acceleration;

- Thermic Accelerometer: they are based on hot gas inside a cavity. Its temperature is monitored by thermopiles. Their most important characteristic is the absence of usury because there are not movable mechanical parts;

- Laser Accelerometer: they are based on a laser interferometer and are characterized by high linearity and low noise floor.

3.2 Project Specifications

Accelerometers for airbag applications need of high speed and accurate response allowing car control unit to make the right choice saving the driver life. For that reason it is important to study a car crash in order to define the specifications of our project. Our aim is the creation of mono axial accelerometer that can be used for both front and lateral car crashes.

3.2.1 Car Crash Modeling

For our application we consider a car crash into a wall because it is the simplest example to define the most important parameters for the project. Another reason is that a lateral crash has a threshold lower than a front car crash so to define the input acceleration range we refer to a front one. Our starting point is to consider a car that travels at constant velocity V_0 equal to 60 km/h. Considering a stop time t_S of 1/10 second and a stop space S_s of 1 meter we have:

$$S = t \cdot V_0 - \frac{1}{2} \cdot a \cdot t^2$$

Thus:

$$S_S = t_S \cdot V_0 - \frac{1}{2} \cdot a \cdot {t_S}^2$$

We get an average acceleration of:

$$a = 2 \cdot \frac{S_S - t_S \cdot V_0}{t_S^2}$$

It corresponds a number of g equal to:

$$n_g = 2 \cdot \frac{S_S - t_S \cdot V_0}{t_S^2} \cdot \frac{1}{g_0}$$

Substituting we get:

$$n_g = 17\ g$$

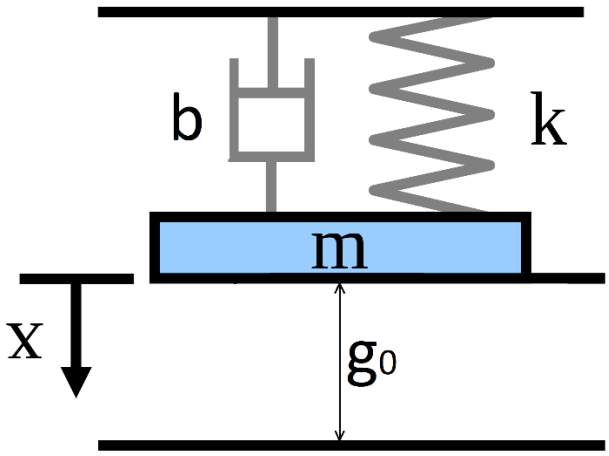
As we can see the acceleration is very high so our MEMS accelerometer needs a big range of sensing. In particular, from a research on web, we found that typical airbag acceleration threshold for a crash is about 8 g and the time between crash and airbag opening is approximately 100 ms. In particular, it is divided into 20 ms where the car control unit makes the decision and 80 ms to the mechanical open of the airbag. Starting from this point, accelerometer

must send a correct value within about 15 ms allowing a correct sampling of signal in the remaining time. In reality a crash it is more complex because it depends on the type of vehicle, on its materials and the dynamic of crash. To simplify our project we consider the following specifications:

- Acceleration threshold value: 8 g;
- Acceleration input range higher than 8 g;
- Rise Time< 10 ms;
- Output Signal compatibility for car control unit (3.3 V, 5 V typical value);

3.3 Modeling of Capacitive Accelerometer

The model of a basic accelerometer is the mass-spring-dashpot mechanical system showed in the figure below.



We consider the positive x axis direct toward the bottom and a starting point of g_0. To write the physical equation we consider that the zero of our reference system is on the mass and their dimension are negligible with respect the x variation. We can write the following relation:

$$m \cdot g'' = m \cdot a_{ext} - k \cdot g - b_0 \cdot g' \quad (1)$$

Where g is the variation from the equilibrium point g_0.

3.3.1 Frequency and Time Responses

Starting from the model equation:

$$m \cdot g'' = m \cdot a_{ext} - k \cdot g - b_0 \cdot g'$$

We can write the Laplace's transform:

$$m \cdot s^2 \cdot X(s) = U(s) - k \cdot X(s) - b_0 \cdot s \cdot X(s)$$

Where X(s) is g and U(s) is $m \cdot a_{ext}$.

The transfer function can be written as:

$$F(s) = \frac{X(s)}{U(s)} = \frac{1/m}{s^2 + \frac{b_0}{m} \cdot s + \frac{k}{m}}$$

Now we can define the main parameters of a second order system:

1) Quality Coefficient Q:

$$Q = \frac{\omega_n \cdot m}{b_0}$$

2) Damping Ratio:

$$\zeta = \frac{1}{2 \cdot Q} = \frac{b_0}{2 \cdot \sqrt{m \cdot k}}$$

3) Undumped Natural Frequency:

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

Defining b, k and m it is possible to draw module and phase Bode diagrams.

For the time step response we can define the following parameters.

1) Overshoot:

$$\mathrm{OV} = \frac{Y_{Max} - Y(\infty)}{Y(\infty)} = \mathrm{e}^{-\frac{\pi \cdot \zeta}{\sqrt{1-\zeta^2}}}$$

2) Rise-Time from 10 % to 90 % of final value:

$$t_r = \frac{1}{\omega_n \cdot \sqrt{1-\zeta^2}} \cdot (\pi - \arccos(\zeta))$$

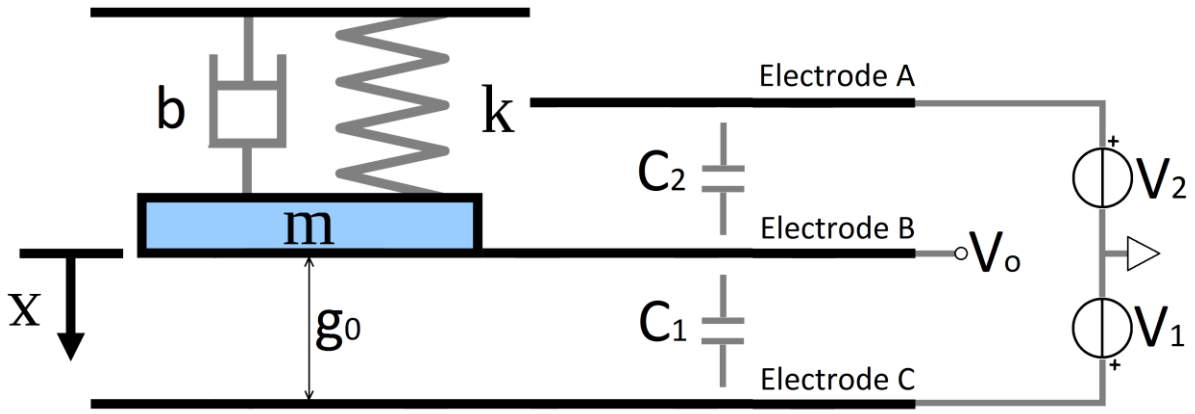
3) Settling-Time within α percentage:

$$t_{st} = -\frac{\ln(\alpha)}{\omega_n \cdot \zeta}$$

These set of parameters are useful for designing our accelerometer and will be used in the following paragraphs inside the project section.

3.3.2 Electrostatic Forces

In order to measure both positive and negative acceleration the conversion of the position into a capacitance is obtained using the following system.



As we can see the system includes two bias voltage generators, V_1 and V_2 connected to two electrodes. The electrode B is the movable plate connected to mechanical system and it corresponds to our electrical output signal. The two voltage generators in fact, are useful to observe the capacitance variations but we have to take in account the electrostatic forces. The relation (1) becomes:

$$m \cdot g'' = m \cdot a_{ext} - k \cdot g - b_0 \cdot g' + F_{el} \quad (2)$$

Where:

$$F_{el} = F_{el1} - F_{el2} \quad (3)$$

$$F_{el1} = \frac{\varepsilon_0 \cdot A \cdot V_1^2}{2 \cdot (g_0 - g)^2} \quad (4)$$

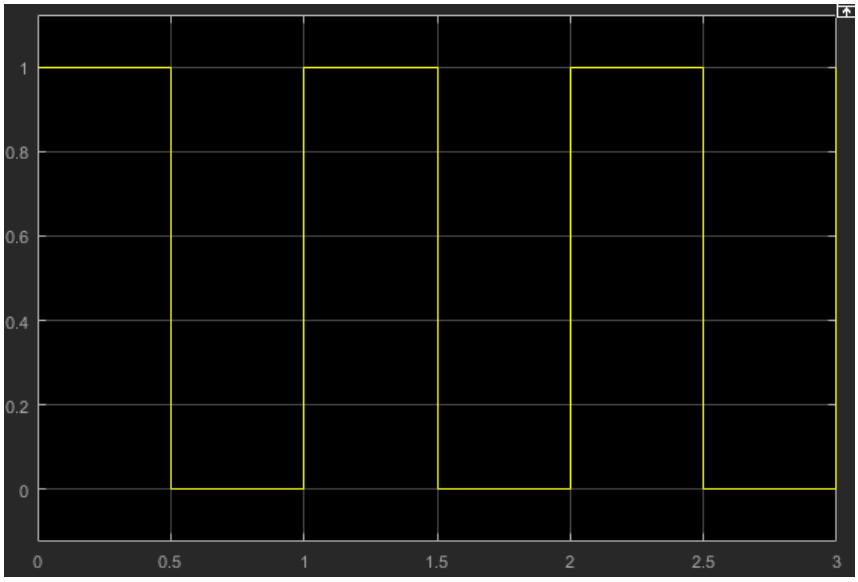
$$F_{el2} = \frac{\varepsilon_0 \cdot A \cdot V_2^2}{2 \cdot (g_0 + g)^2} \quad (5)$$

If we consider that V_1 and V_2 have a frequency much higher than the cut frequency of the mechanical system, we can use the average value of the quadratic of the signal generators. Our choice is to use two pulsed signals that can be easily generated inside a digital system or in the final integrated circuit. If we consider that:

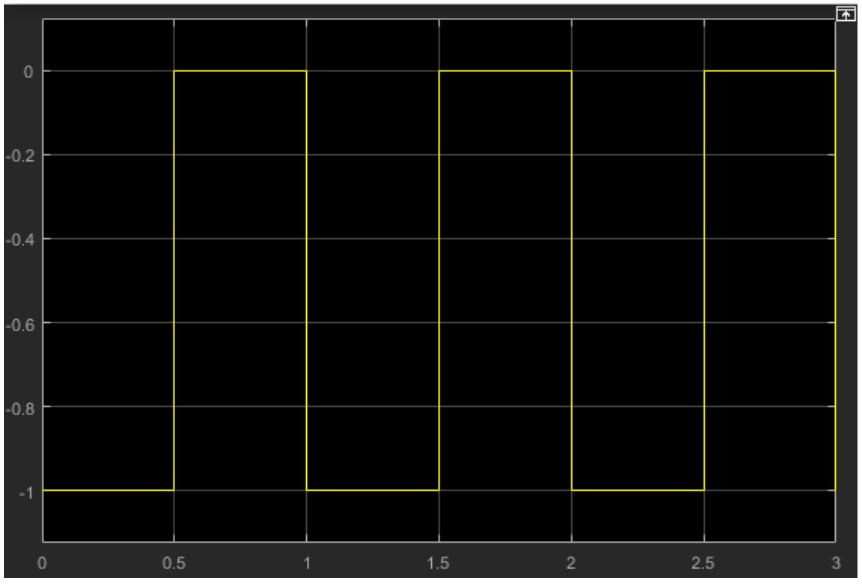
$$V_1 = \frac{V_P}{2} \cdot \text{Pulse}(t) + \frac{V_P}{2}$$

$$V_2 = -\frac{V_P}{2} \cdot \text{Pulse}(t) - \frac{V_P}{2}$$

Example of V_1 signal



Example of V_2 signal



The average value of the quadratic signals is:

$$\overline{V_1}^2 = \overline{V_2}^2 = \frac{V_P^2}{2}$$

So the electrostatic forces become:

$$F_{el1} = \frac{\varepsilon_0 \cdot A \cdot V_1^2}{2 \cdot (g_0 - g)^2} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4 \cdot (g_0 - g)^2}$$

$$F_{el2} = \frac{\varepsilon_0 \cdot A \cdot V_2^2}{2 \cdot (g_0 + g)^2} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4 \cdot (g_0 - g)^2}$$

The resultant force is:

$$F_{el} = F_{el1} - F_{el2} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4}\left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2}\right]$$

The relation (2) becomes:

$$m \cdot g'' = m \cdot a_{ext} - k \cdot g - b \cdot g' + \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4}\left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2}\right]$$

3.3.3 Damper Coefficient

Inside our model we have considered the damper coefficient b as a constant. Due to the air enclosed inside MEMS and for large acceleration values, b becomes a function of the deflection and it can be represented by the following formula:

$$b = \frac{1}{2} \cdot u \cdot A^2 \cdot \left(\frac{1}{(g_0 - g)^3} + \frac{1}{(g_0 + g)^3}\right)$$

Where:

- u is the air viscosity equal to $1.81 \cdot 10^{-5} \frac{kg}{m \cdot s}$;
- A is the area between the plates;
- g_0 is the start value of the displacement g;

We define the initial value of b as: $b_0 = \frac{u \cdot A^2}{g_0^3}$

So the formula (2) becomes:

$$m \cdot g'' = m \cdot a_{ext} - k \cdot g - \frac{1}{2} \cdot u \cdot A^2 \left(\frac{1}{(g_0 - g)^3} + \frac{1}{(g_0 + g)^3}\right) g' + \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4}\left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2}\right] \quad (3)$$

3.3.4 Brownian Noise

Noise is an important parasitic effect that can be observed in each physical system. In particular we consider the thermal-mechanical noise called Brownian noise. It creates a random force due to Brownian motion of air molecules caused by damping. It is direct proportional to the damper coefficient and creates an equivalent acceleration applied to the seismic mass. Its formula is:

$$a_{Bw} = \sqrt{\frac{4 \cdot k_B \cdot T \cdot \omega_n}{m \cdot Q}} \frac{m/s^2}{\sqrt{Hz}}$$

Where:

- ω_n is the undamped natural frequency;
- Q is the quality factor;
- m is the seismic mass;
- T is the absolute temperature in Kelvin;
- k_B is Boltzmann coefficient equal to $1.3806488 \cdot 10^{-23} m^2 \cdot kg \cdot s^{-2} \cdot K^{-1}$;

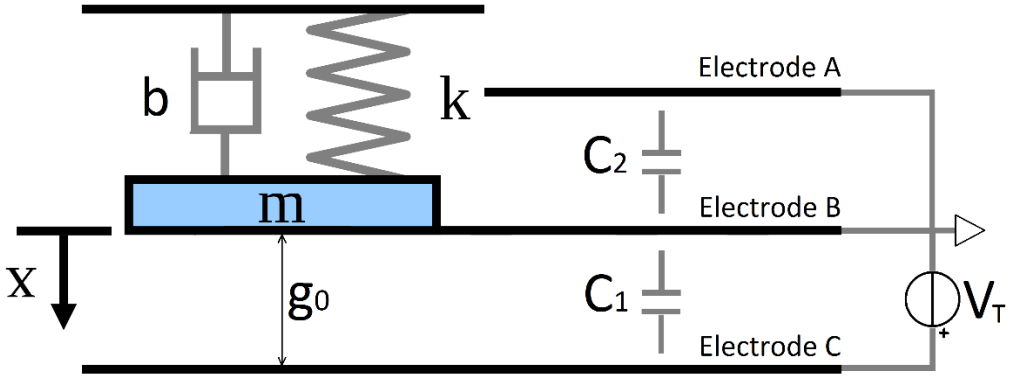
Brownian noise affects the sensing accuracy and it must be limited. Typical values are near 0.1 $mg/\sqrt{Hz}$. The equivalent acceleration is added to the input acceleration to take into account the noise effect. Considering also the Brownian noise inside the system equation the formula (3) becomes:

$$m \cdot g'' = m \cdot (a_{ext} + a_{BW}) - k \cdot g - \frac{1}{2} \cdot u \cdot A^2 \left(\frac{1}{(g_0 - g)^3} + \frac{1}{(g_0 + g)^3} \right) g' + \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4} \left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2} \right] \quad (4)$$

Where the a_{BW} is the acceleration due to the Brownian noise.

3.3.5 Pull-In Voltage

In first approximation we can consider b as constant value and define the pull-in voltage of our accelerometer. In particular we consider no external forces and only the action of V_T. In this way electrodes A and B are at zero potential. The circuit for the pull-in voltage is showed below:



The resulting force becomes:

$$F_{Tot} = -k \cdot g + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{2} \cdot \left[\frac{1}{(g_0 - g)^2}\right]$$

Now we consider the first derivative of F_{Tot}.

$$\frac{dF_{Tot}}{dg} = -k + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{1} \cdot \left[\frac{1}{(g_0 - g)^3}\right]$$

Since we want that for a positive dg the resulting dF_{Tot} becomes negative to have a negative feedback we impose that:

$$\frac{dF_{Tot}}{dg} = -k + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{1} \cdot \left[\frac{1}{(g_0 - g)^3}\right] < 0$$

Considering the upper limit we have:

$$\frac{dF_{Tot}}{dg} = -k + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{1} \cdot \left[\frac{1}{(g_0 - g)^3}\right] = 0$$

So we obtain k_{PI}:

$$k_{PI} = \frac{\varepsilon \cdot A \cdot V_T^2}{(g_0 - g)^3}$$

Now substituting k_{PI} in F_{Tot}=0 we found:

$$F_{Tot} = -k_{Pi} \cdot g + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{2} \cdot \left[\frac{1}{(g_0 - g)^2}\right] = 0$$

Thus:

$$F_{Tot} = -\frac{\varepsilon_0 \cdot A \cdot V_T^2}{(g_0 - g)^3} \cdot g + \frac{\varepsilon_0 \cdot A \cdot V_T^2}{2} \cdot \left[\frac{1}{(g_0 - g)^2}\right] = 0$$

$$-g + \frac{(g_0 - g)}{2} = 0$$

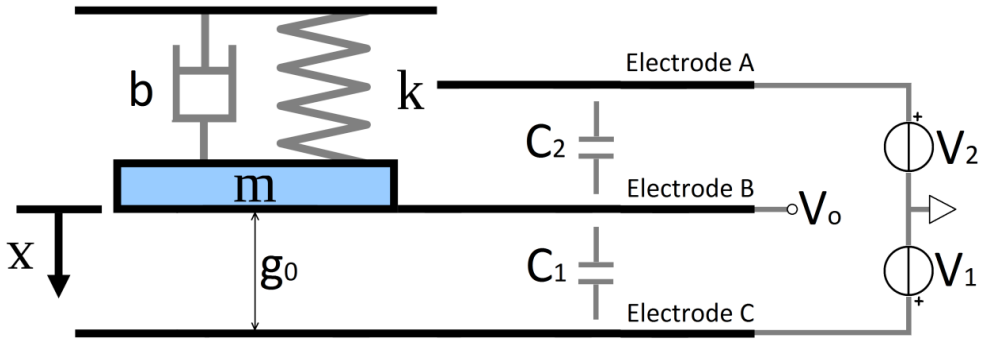
$$g_{PI} = \frac{g_0}{3}$$

We have found g_{PI} and from k_{PI} and so the pull-in voltage is:

$$k_{PI} = \frac{\varepsilon_0 \cdot A \cdot V_{PI}^2}{(g_0 - g)^3} = \frac{\varepsilon_0 \cdot A \cdot V_{PI}^2}{\left(g_0 - \frac{g_0}{3}\right)^3}$$

$$V_{PI} = \sqrt{\frac{8 \cdot k_{PI} \cdot g_0^3}{27 \cdot \varepsilon_0 \cdot A}}$$

The same result can be obtained if we consider V_T applied between A and B electrodes. Normally C_1 is equal to C_2 at the equilibrium position and remembering that the system is:



For any value of V_1 and V_2 the resultant electrostatic force is zero so we do not have to consider the pull-in effect.

3.3.6 Maximum Acceleration

Talking of maximum acceleration is important to define the input range of our system. We consider that the g variation is between $-g_0 + g_{min}$ and $g_0 - g_{min}$. These limits strongly depend on the mechanical realization of the accelerometer. In any case the mobile plate cannot collides into one of the fixed plates so the displacement variation must be considered limited. We neglect the effects of electrostatic forces and damper coefficient variation to find a theoretical value of the maximum acceleration. Imposing the value of g equal to $g_0 - g_{min}$, we find the maximum acceleration that we can apply towards the mobile plate B.

Imposing that for $g = g_0 - g_{min}$ the resulting force becomes zero:

$$0 = m \cdot a_{ext+} - k \cdot (g_0 - g_{min})$$

We have the maximum positive acceleration:

$$a_{ext+} = \frac{k \cdot (g_0 - g_{min})}{m}$$

Imposing that for $g = -g_0 + g_{min}$the resulting force becomes zero:

$$0 = m \cdot a_{ext-} - k \cdot (-g_0 + g_{min})$$

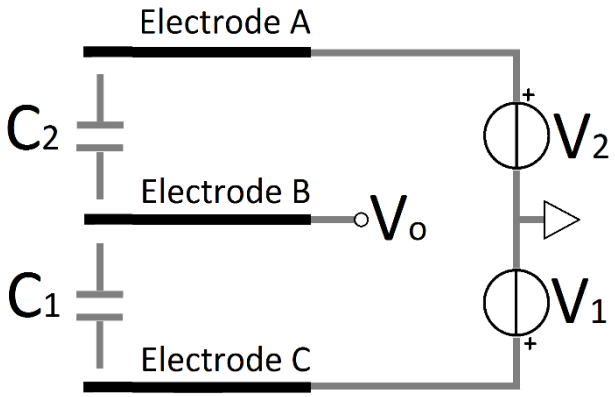
We have the maximum negative acceleration:

$$a_{ext-} = \frac{k \cdot (-g_0 + g_{min})}{m}$$

In reality because of the electrostatic forces and the damper coefficient variation, these maximum values are only theoretical and the real maximum acceleration is a bit smaller for both negative and positive maximum displacement. During the project is important to minimize the variation of damper coefficient and the maximum value of the resulting electrostatic force when the maximum acceleration is applied.

3.3.7 Output Function

The output function represents the relationship between the variation of the capacitances ΔC and the output voltage V_O. We consider the following equivalent electrical circuit:



Applying the superimposition principle, we can calculate the output function as follow. Considering $V_2=0$, we have:

$$V'_{Out}(s) = \frac{\frac{1}{s \cdot C_2}}{\frac{1}{s \cdot C_2} + \frac{1}{s \cdot C_1}} \cdot V_1 = \frac{s \cdot C_1}{s \cdot C_1 + s \cdot C_2} \cdot V_1$$

Considering $V_1=0$, we have:

$$V''_{Out}(s) = \frac{\frac{1}{s \cdot C_1}}{\frac{1}{s \cdot C_2} + \frac{1}{s \cdot C_1}} \cdot V_2 = \frac{s \cdot C_2}{s \cdot C_1 + s \cdot C_2} \cdot V_2$$

Thus:

$$V_{Out}(s) = V'_{Out}(s) + V''_{Out}(s) = \frac{(V_1 \cdot C_1 + V_2 \cdot C_2)}{(C_1 + C_2)}$$

Calling C_0 the value of the two capacitors when the g variation is zero (zero external acceleration) we can write that $C_1 = C_0 + \Delta C$ and $C_2 = C_0 - \Delta C$. Since $V_1=-V_2$ we obtain:

$$V_{Out}(s) = \frac{\Delta C \cdot V_1(s)}{C_0}$$

The resulting gain is:

$$\frac{V_{Out}(s)}{\Delta C \cdot V_1(s)} = \frac{1}{C_0}$$

The last relation shows the importance of the starting value of the capacitors because it is the gain of our system.

The capacitors can be calculated as a function of the displacement g:

$$C_1 = \frac{\varepsilon_0 \cdot A}{(g_0 - g)}$$

And:

$$C_2 = \frac{\varepsilon_0 \cdot A}{(g_0 + g)}$$

At zero g displacement C_1 and C_2 are equal to:

$$C_0 = \frac{\varepsilon_0 \cdot A}{g_0}$$

From these relations we can write the capacitance C_1 and C_2 as:

$C_1 = C_0 + \Delta C$ and $C_2 = C_0 - \Delta C$.

We have that:

$$C_1 - C_2 = 2 \cdot \Delta C = \frac{\varepsilon_0 \cdot A}{(g_0 - g)} - \frac{\varepsilon_0 \cdot A}{(g_0 + g)} = \frac{\varepsilon_0 \cdot A \cdot g \cdot 2}{g_0^2 - g^2}$$

And we finally optain:

$$\Delta C = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}$$

3.3.8 Final Model

At this point we have created a complete model of our accelerometer and we can represent it with the following state equations.

Choosing g and g' as state variables we obtain:

$$\begin{cases} X_1 = g \; ; X_{1,0} = g_0 \\ \quad X_2 = g' \end{cases}$$

$$\begin{cases} X_1' = X_2 \\ X_2' = \frac{1}{m} \cdot \left(m \cdot (a_{ext} + a_{Bw}) - k \cdot X_1 - \frac{1}{2} \cdot u \cdot A^2 \cdot \left(\frac{1}{(X_{1,0} - X_1)^3} + \frac{1}{(X_{1,0} + X_1)^3} \right) \cdot X_2 - \frac{\varepsilon \cdot A \cdot V_p^2}{4} \cdot \left[\frac{1}{(X_{1,0} - X_1)^2} - \frac{1}{(X_{1,0} + X_1)^2} \right] \right) \end{cases}$$

And:

$$V_{Out}(s) = \frac{\Delta C \cdot V_1(s)}{C_0}$$

Where:

$$a_{Bw} = \frac{\sqrt{4 \cdot k_B \cdot \omega_n}}{m}$$

$$\Delta C = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}$$

$$C_0 = \frac{\varepsilon_0 \cdot A}{g_0}$$

Thanks to this representation we can create the simulation model on Simulink and with VHDL-AMS.

3.4 Design of Accelerometer

Project of the accelerometer starts recalling the specifications:

- Acceleration threshold value: 8 g;
- Rise-Time$<$ 10 ms;
- Output signal compatibility with modern electronic circuit (3.3 V, 5 V typical values).

Modelling a crash acceleration is extremely complex. For this reason we have chosen a step input acceleration because it represents the maximum speed variation obtainable. In order to guarantee a fast step response with a limited number of oscillation we have imposed:

- Overshoot $=$ 0.1 %;
- Rise Time $=$ 2 ms;

From these definitions we can calculate the damping ratio ζ and the undamped natural frequency ω_n. From the overshoot definition:

$$\mathrm{OV} = \frac{Y_{Max} - Y(\infty)}{Y(\infty)} = \mathrm{e}^{-\frac{\pi\cdot\zeta}{\sqrt{1-\zeta^2}}} = 0.001$$

We obtain:

$$\zeta = \sqrt{\frac{\ln(\mathrm{OV})^2}{\pi^2 + \ln(OV)^2}} = 0.91$$

And from:

$$t_r = \frac{1}{\omega_n \cdot \sqrt{1-\zeta^2}} \cdot (\pi - \arccos(\zeta)) = 0.002\ s$$

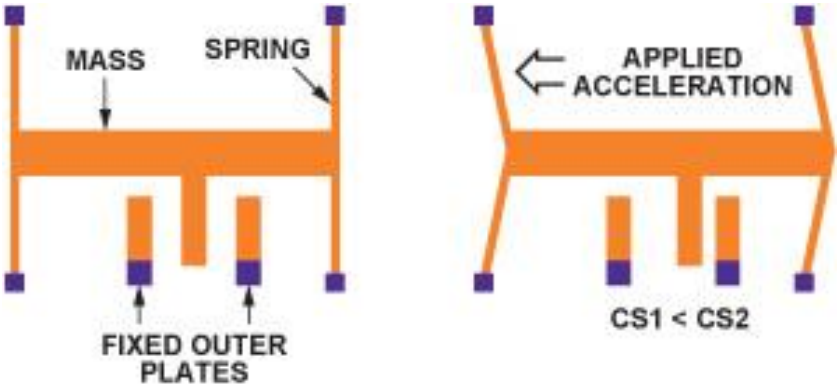
We have:

$$\omega_n = \frac{1}{t_r \cdot \sqrt{1-\zeta^2}} \cdot (\pi - \arccos(\zeta)) = 3279 \frac{\mathrm{rad}}{\mathrm{s}}$$

The Settling-Time within 1 % is:

$$t_{st} = -\frac{\ln(\alpha)}{\omega_n \cdot \zeta} = 1.5\ ms$$

From the transfer function we can calculate m and b_0 imposing k spring constant. The value of k depends on the mechanical structure of the accelerometer. A typical structure is showed in the figure below:



Its material is usually polycrystalline silicon and k depends on fabrication processes, length, width and thickness of the spring section. Typical values are from 1 to 10 N/m. From a research document, we found an example formula:

$$k = \frac{16 \cdot W \cdot t^3}{L^3} \cdot E_R$$

With L= 150 μm, W= 2 μm, t= 2 μm and E_R= 160 GPa we found k equal to about 3 N/m.

Imposing k equal to 3 N/m and remembering that:

$$F(s) = \frac{Y(s)}{X(s)} = \frac{1/m}{s^2 + \frac{b_0}{m} \cdot s + \frac{k}{m}}$$

A) Quality Coefficient Q:

$$Q = \frac{\omega_n \cdot m}{b_0}$$

B) Damping Ratio:

$$\zeta = \frac{1}{2 \cdot Q} = \frac{b}{2 \cdot \sqrt{m \cdot k}}$$

C) Undumped Natural Frequency:

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

$$m = \frac{k}{\omega_n^2} = 2.79 \cdot 10^{-7} kg$$

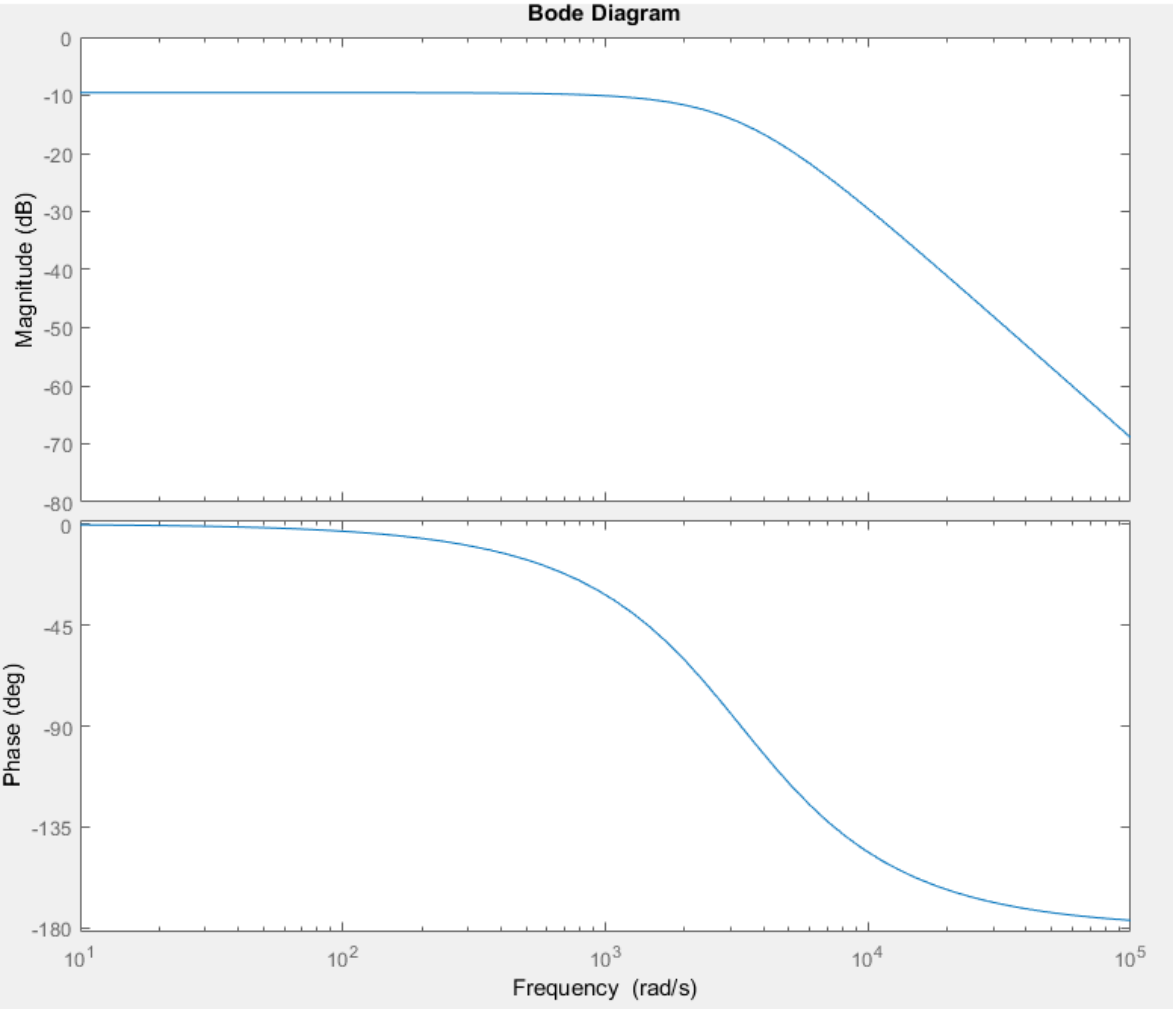
From B):

$$\mathrm{Q} = \frac{1}{2 \cdot \zeta} = 0.55$$

Finally from A):

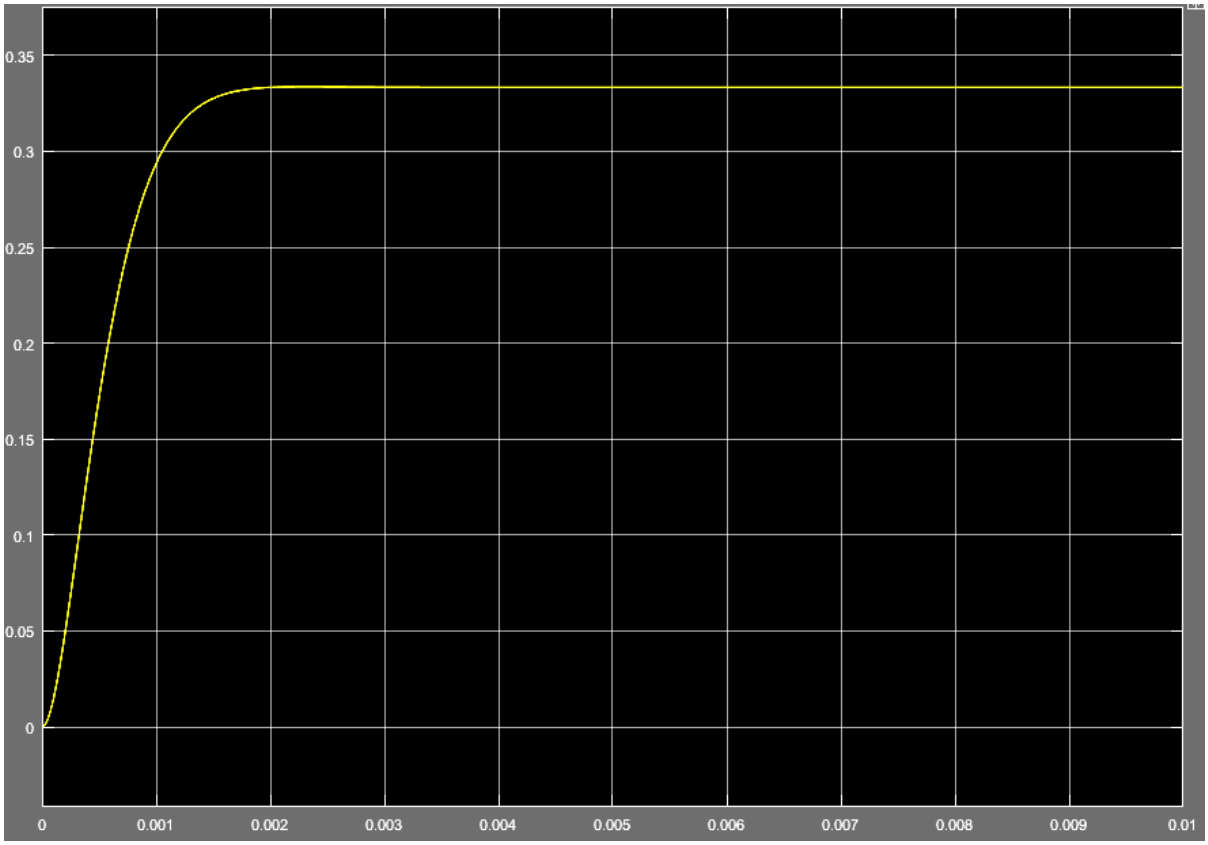
$$b_0 = \frac{\omega_n \cdot m}{Q} = 0.0017\ m \cdot kg$$

The resulting bode diagrams are:



While the step response is:

We obtain from C):



As we can see within 10 ms we have a stable value of the displacement so our specification is respected.

At this point we can calculate from b_0 formula the value of Area A but we must impose g_0. Choosing g_0 equal to 100 μm we obtain:

$$b_0 = \frac{u \cdot A^2}{g_0^3} \rightarrow A = \sqrt{\frac{u}{g_0^3 \cdot b_0}} = 9.5933 \cdot 10^{-6}\, m^2$$

The initial value of capacitances C_1 and C_2 is:

$$C_0 = \frac{\varepsilon_0 \cdot A}{g_0} = 0.849\, pF$$

Note: $\varepsilon_0 = 8.854 \cdot 10^{-12}$.

Imposing g_{min} equal to 90 μm we have a maximum accelerations:

$$a_{ext+} = \frac{k \cdot (g_0 - g_{min})}{m} = 107.5042 \frac{m}{s^2} \rightarrow 10.96\, g$$

$$a_{ext-} = \frac{k \cdot (g_{min} - g_0)}{m} = -107.5042 \frac{m}{s^2} \rightarrow -10.96\, g$$

These values are higher than the airbag threshold acceleration equal to 8 g so the input range is respected.

3.4.1 Voltage Generators V_1 and V_2

Choosing a value of V_P equal to 5 V we can integrate our MEMS in a standard logic integrated circuit. Other possibility could be 3.3 V or 2.5 V, typical voltages in modern integrated circuit. The frequency is chosen equal to 100 kHz, much higher than the natural frequency of our mechanical system.

3.4.2 Output Voltage Range

The calculation of the output voltage range is dependent from the maximum ΔC variations:

$$\Delta C_{max+} = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}\Big|_{g=g_0-g_{min}} = \frac{\varepsilon_0 \cdot A \cdot (g_0 - g_{min})}{g_0^2 - (g_0 - g_{min})^2} = 85.8\, fF$$

So the maximum output signal V_O is:

$$V_{Out}(s)_{max+} = \frac{\Delta C_{max+} \cdot V_1(s)_{max}}{C_0} = 0.505\, V$$

This value corresponds to an external acceleration equal to 10.96 g. For an external acceleration of -10.96 g we have:

$$\Delta C_{max-} = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}\Big|_{g=g_{min}-g_0} = \frac{\varepsilon_0 \cdot A \cdot (-g_0 + g_{min})}{g_0^2 - (-g_0 + g_{min})^2} = -85.8\, fF$$

$$V_{Out}(s)_{max-} = \frac{\Delta C_{max-} \cdot V_1(s)_{max}}{C_0} = -0.505\, V$$

Neglecting the electrostatic forces, damper coefficient variation and Brownian noise we can calculate the g variation at 8 g acceleration:

From:

$$m \cdot g'' = m \cdot a_{8g} - k \cdot g - b_0 \cdot g'$$

We obtain:

$$g_{8g} = \frac{m \cdot a_{8g}}{k} = 7.3\ \mu m$$

So:

$$\Delta C_{8g} = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}\Big|_{g_{8g}} = \frac{\varepsilon_0 \cdot A \cdot g_{8g}}{g_0^2 - g_{8g}^2} = 62.3\, fF$$

That corresponds to:

$$V_{Out}(s)_{max} = \frac{\Delta C_{8g} \cdot V_1(s)_{max}}{C_0} = 0.367\ V$$

The same results can be obtained with an acceleration of -8 g.

3.4.3 Electrostatic Forces Effect

The electrostatic forces effect depends on the displacement g:

$$F_{el} = F_{el1} - F_{el2} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4}\left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2}\right]$$

We can evaluate the maximum values of F_{el} at $g = g_0 - g_{min}$ and $g = -g_0 + g_{min}$. In particular we calculate the equivalent accelerations of electrostatic forces:

$$A_{el+} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4 \cdot m}\left[\frac{1}{(g_0 - g_0 + g_{min})^2} - \frac{1}{(g_0 + g_0 - g_{min})^2}\right] = 0.0776\frac{m}{s^2} \rightarrow 0.0079\ g$$

$$A_{el-} = \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4 \cdot m}\left[\frac{1}{(g_0 + g_0 - g_{min})^2} - \frac{1}{(g_0 - g_0 + g_{min})^2}\right] = -0.0776\frac{m}{s^2} \rightarrow -0.0079\ g$$

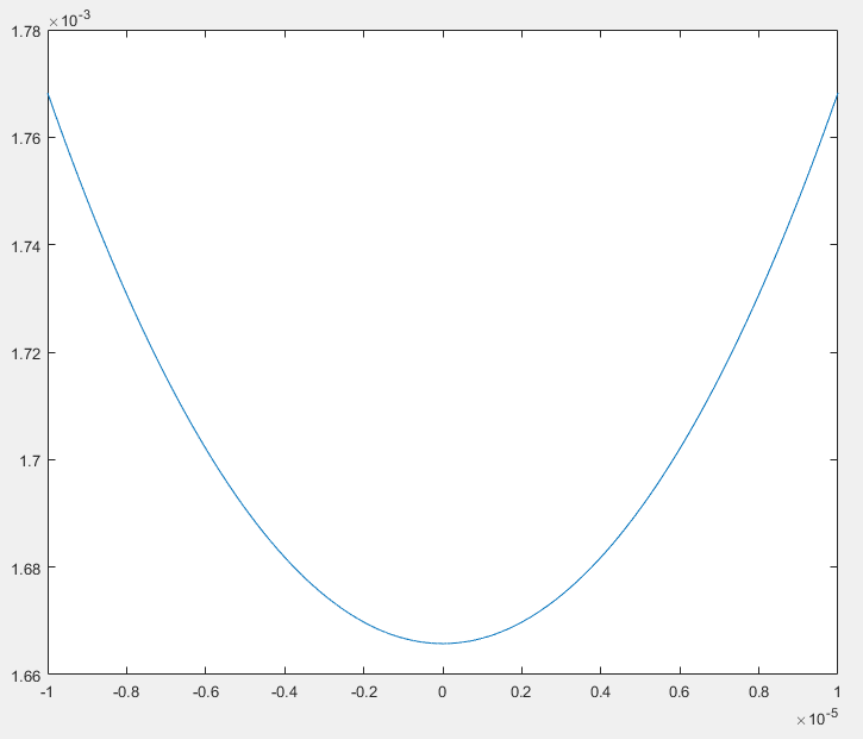
These accelerations are coherent with respect to the external one. Their values are much smaller than the maximum acceleration equal to 10.96 g. From this analysis we can consider a good choice the values assigned to A, m, V_P, g_0 and g_{min}.

3.4.4 Damping Variation Effect

Damping variation effect is important because a variation of b changes the frequency and time response of the system. For this reason we want to minimize its variation. Recalling the formula:

$$b = \frac{1}{2} \cdot u \cdot A^2 \cdot \left(\frac{1}{(g_0 - g)^3} + \frac{1}{(g_0 + g)^3} \right)$$

We can plot its behavior as a function of g variation between $g_0 - g_{min}$ and $-g_0 + g_{min}$. The graph results:



As we can see the variation of b around b_0 value is small and it is a good result.

The percentage of the variation is equal to:

$$b\% = 100 \cdot \left(1 - \frac{b_0}{b_{max}}\right) = 100 \cdot \left(1 - \frac{0.001700}{0.001768}\right) = 5.8\ \%$$

3.4.5 Brownian Noise

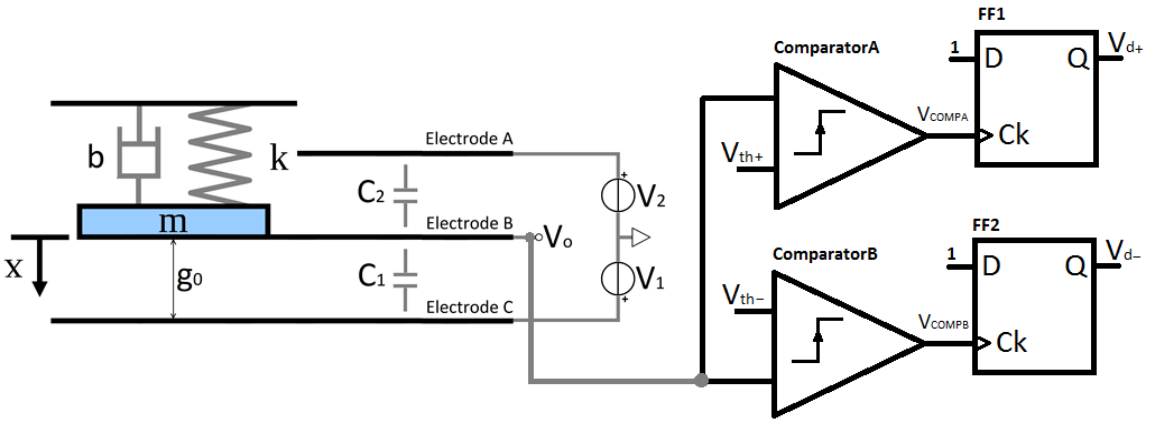
After the designing we can evaluate the value of Brownian noise at T=298 K:

$$a_{Bw} = \sqrt{\frac{4 \cdot k_B \cdot T \cdot \omega_n}{m \cdot Q}} = 1.87 \cdot 10^{-5} \frac{m}{s^2 \cdot \sqrt{Hz}} \rightarrow 1.91 \cdot 10^{-6} \frac{g}{\sqrt{Hz}}$$

Its value limits the smallest acceleration that our accelerometer can detect.

3.4.6 Output Circuit

The output circuit is a block that generates a digital signal if the acceleration threshold of 8 g is exceeded for both positive and negative ones. Since the V_O at 8 g has a value of 0.367 V we can use an analog **ComparatorA** with a threshold V_{th+} of 0.367 V. The same for the negative acceleration of −8 g using the **ComparatorB**. In this case we have to use a negative threshold V_{th-} of −0.367 V. The V_{COMPA} and V_{COMPB} will be 1 logic when the thresholds will be exceeded and so the corresponding D Flip-Flop will be set. We have chosen to use two comparators because in this way the accelerometer can be mounted on the final target in two possible direction. In a more general analysis the two threshold values can also be different. The circuit is the following:



In this way as soon as one of the acceleration thresholds are exceeded the output of one comparator will be at '1' logic. In this way a theoretical digital circuit can use it as an interrupt or a digital input.

3.4.7 MatLab Scripts Designing

Every passage of our design has been done with MatLab through some scripts. This project approach allowed us to calculate each parameters in a simple way.

Script 1

```
function [smorzamento,wn,tass] = design(overshoot,tr,alpha)
smorzamento=sqrt((log(overshoot/100)^2)/(pi^2+log(overshoot/100)^2))
wn=1/(tr*sqrt(1-smorzamento^2))*(pi-acos(smorzamento))
tass=-log(alpha/100)/wn/smorzamento
end
```

Script 2

```
function [num,den,m,b,Q] = dodesign(smorzamento,wn,k,tass)
close all;
m=k/wn^2;
Q=1/2/smorzamento;
b=wn*m/Q
t=linspace(0,3*tass,10000);
s=tf('s');
f=(1/m)/(s^2+b/m*s+k/m);
[num,den]=tfdata(f,'v');
figure
bode(f,{10, 100000})
end
```

Design Script

```
clear all;
close all;
%---------------Constatns-----------------
u0=1.81e-5;
k_b=1.3806488e-23;
e=8.854e-12;
k=3;
 %--------Overshoot And RiseTime definitions--------
overshoot=0.1;  % 0.1%
risetime=0.002  ;   % 0.002 secondi
 [smorzamento,wn,tass] = design(overshoot,risetime,5);
[num,den,m,b0,Q] = dodesign(smorzamento,wn,k,tass)
Ampiezza=5;
g0=100e-6;
g_min=90e-6;
A=sqrt(g0^3*b0/u0)
C0=(e*A)/g0
Gain=20*log10(1/C0)
a_max=k*(g0-g_min)/m
a_min=k*(-g0+g_min)/m
 a_elMax=-(e*A*(Ampiezza^2)/4/m)*(1/(g_min^2)-1/(2*g0-g_min)^2)
a_elMin=-(e*A*Ampiezza^2/4/m)*(-1/g_min^2+1/(2*g0-g_min)^2)
VPull=sqrt(16*k*g0^3/27/e/A)
x=linspace(-g0+g_min,g0-g_min,10000);
figure
plot(x,1/2*u0*A^2.*( 1./(g0-x).^3+1./(g0+x).^3))
 bmax=1/2*u0*A^2*( 1/(g0-(g0-g_min))^3+1/(g0+(g0-g_min))^3)
deltab=100-b0/bmax*100
a=a_max;
f=wn/2/pi
ThresholdAirbag=0.367;  %mv
```

3.5 Simulation: Simulink

3.5.1 Simulink

Simulink is an internal MatLab tool. In particular, it is a block diagram environment for multi-domain simulation and designing. Simulink also provides the possibility to generate code automatically, continuous test and verification of embedded systems. It has a graphical editor with customizable libraries. There are a lot of solvers for the modelling and the simulation of dynamic systems. Another important possibility is the integration inside Matlab enabling you to incorporate algorithms into models and export simulation results to MatLab for post-processing analysis.

3.5.2 From The Mathematical Model To The Simulink Model

Starting from the model equations:

$$\begin{cases} X_1 = g\ ; X_{1,0} = g_0 \\ \quad X_2 = g' \end{cases}$$

$$\begin{cases} X_1' = X_2 \\ X_2' = \frac{1}{m} \cdot \left(m \cdot (a_{ext} + a_{Bw}) - k \cdot X_1 - \frac{1}{2} \cdot u \cdot A^2 \cdot \left(\frac{1}{(X_{1,0} - X_1)^3} + \frac{1}{(X_{1,0} + X_1)^3} \right) \cdot X_2 - \frac{\varepsilon \cdot A \cdot V_P^2}{4} \cdot \left[\frac{1}{(X_{1,0} - X_1)^2} - \frac{1}{(X_{1,0} + X_1)^2} \right] \right) \end{cases}$$

$$V_{Out}(s) = \frac{\Delta C \cdot V_1(s)}{C_0}$$

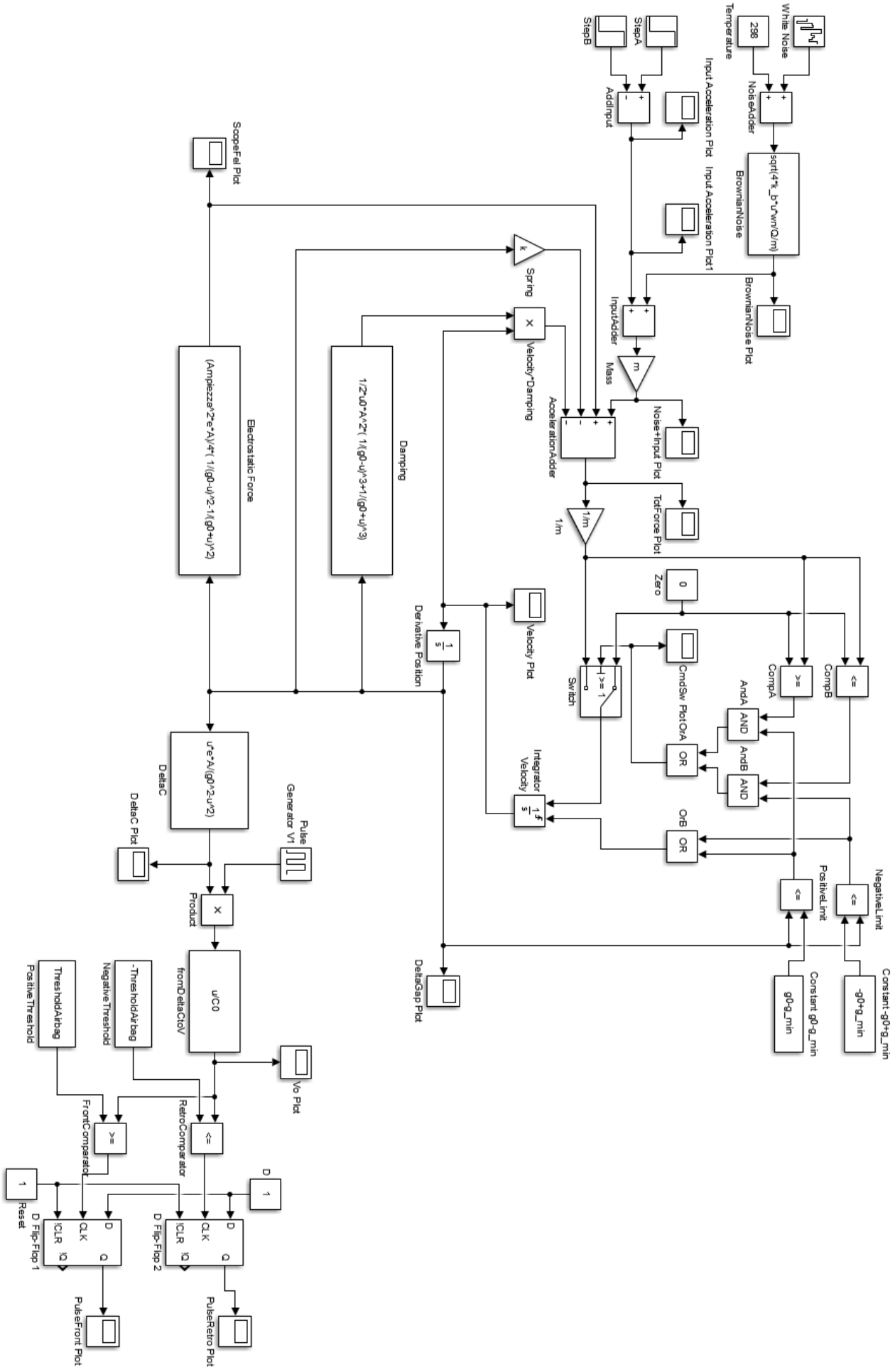
$$a_{Bw} = \frac{\sqrt{4 \cdot k_B \cdot \omega_n}}{m}$$

$$\Delta C = \frac{\varepsilon_0 \cdot A \cdot g}{g_0^2 - g^2}$$

$$C_0 = \frac{\varepsilon_0 \cdot A}{g_0}$$

We can create the simulation model inside Simulink. All parameters are define in MatLab workspace through the previous scripts.

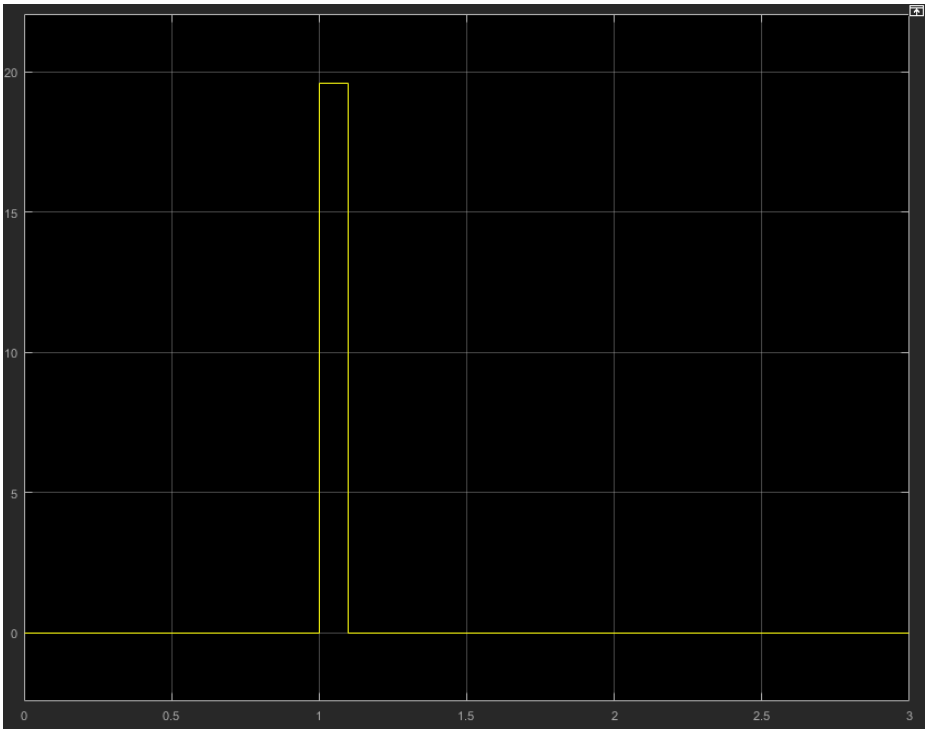
The Simulink model is showed below:



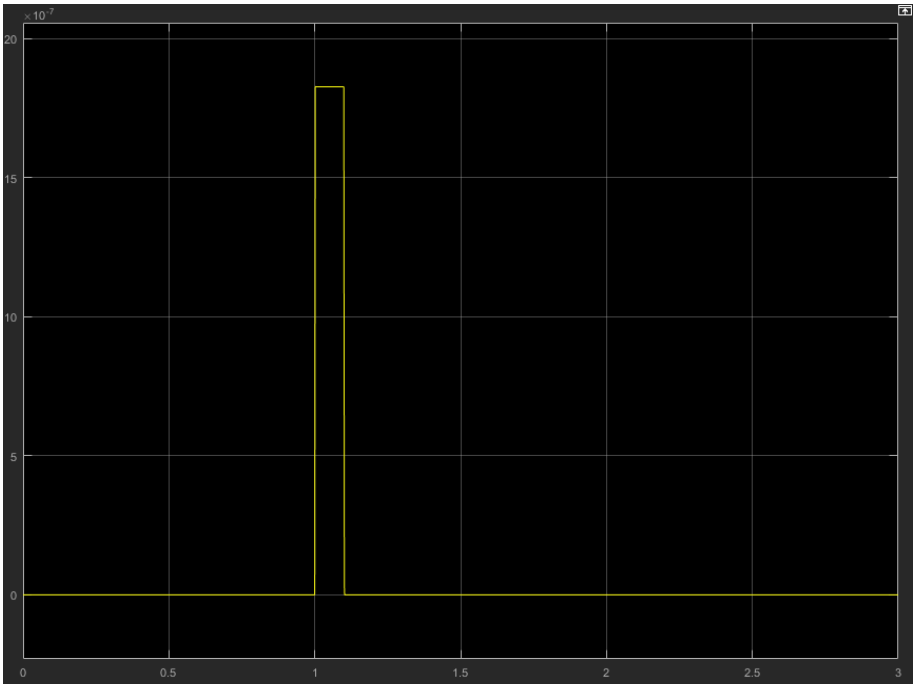
As we can see, there are a lot of blocks inside the model. On the left we have the input step acceleration summed with the Brownian noise through the **InputAdder**. Noise is created by the block **BrownianNoise.** The input temperature is created by the sum of a white noise generator and a constant **Temperature.** In this way the resulting temperature has an average value of 298 K plus a time variation of ± 0.1 K. The resulting input acceleration is multiplied by the mass m through the **Mass** block. The next block is **AccelerationAdder.** This adder sums the input acceleration, the spring force, the dashpot force and resultant electrostatic force. The resultant force is applied to our mobile plate divided by the mass and integrated by **Integrator Velocity** obtaining the velocity. Finally we obtain the position variation g from **Derivative Position** block. The displacement g is used by **Damping**, **Electrostatic Acceleration** and **Spring** blocks. As we can see the resultant acceleration passes through the **Switch** block. It allows with some other blocks the saturation of displacement g to $g_0 - g_{min}$ for a positive resultant acceleration and to $-g_0 + g_{min}$ for a negative one. The comparators **PositiveLimit** and **NegativeLimit** give us an output equal to '1' logic when g exceeds one of the two limits. In these cases the **OrB** resets the **IntegratorVelocity** block in order to force the velocity to a zero value. When we have the positive saturation $g = g_0 - g_{min}$ and the resultant acceleration continues to be positive we have to force it to zero so the **CompA** in AND with the **PositiveLimit** block, forces the resultant acceleration to zero through the **Switch** block. Otherwise if the resultant acceleration changes its sign the system works normally. The same thing happens when g is equal to the negative limit. In fact **CompB** in AND with **NegativeLimit** block forces the acceleration to zero and if it continues to be negative. Finally we use the displacement g to calculate ΔC through **DeltaC** block. The output is multiplied by the signal generator **PulseGenerator** that represents the generators V_1 and V_2. The resultant output is applied to **fromDeltaCtoV** block in order to obtain the final V_O signal. The output signal is used by the two comparator **RetroComparator** and **FrontComparator** with two D Flip-Flops to detect the exceeding of the positive or the negative airbag threshold. Testing involves in six simulations for both positive and negative external accelerations. In the following paragraphs the results are showed.

3.5.3 Simulation: Positive Acceleration

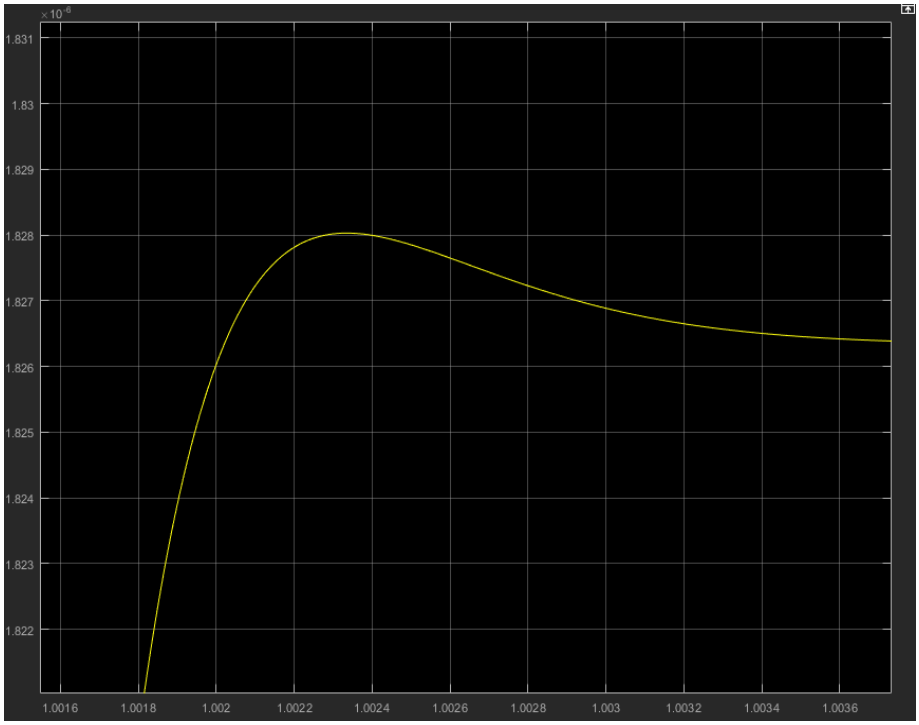
The positive Acceleration is set to 2 g and we have used a step input signal showed in the figure below.



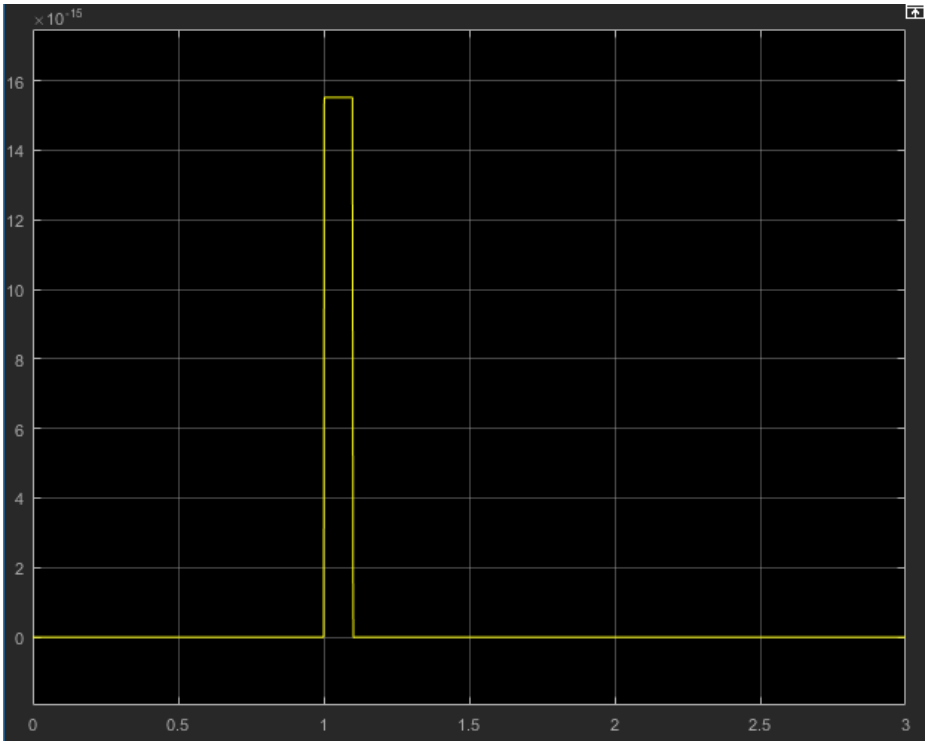
The resulting displacement g is:



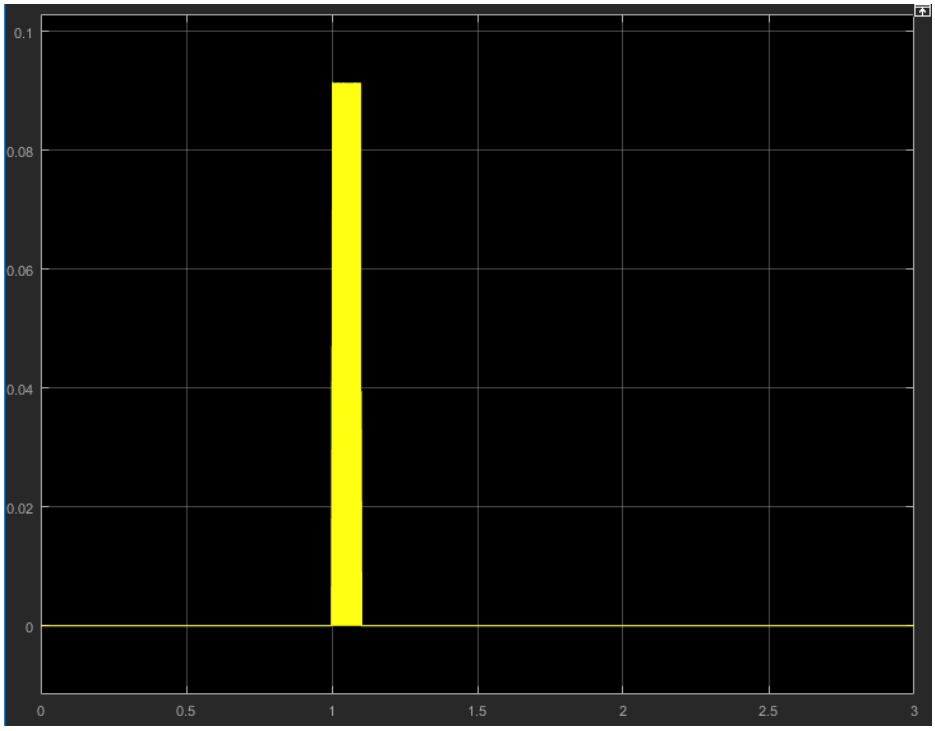
If we zoom the graph, the displacement behavior is:



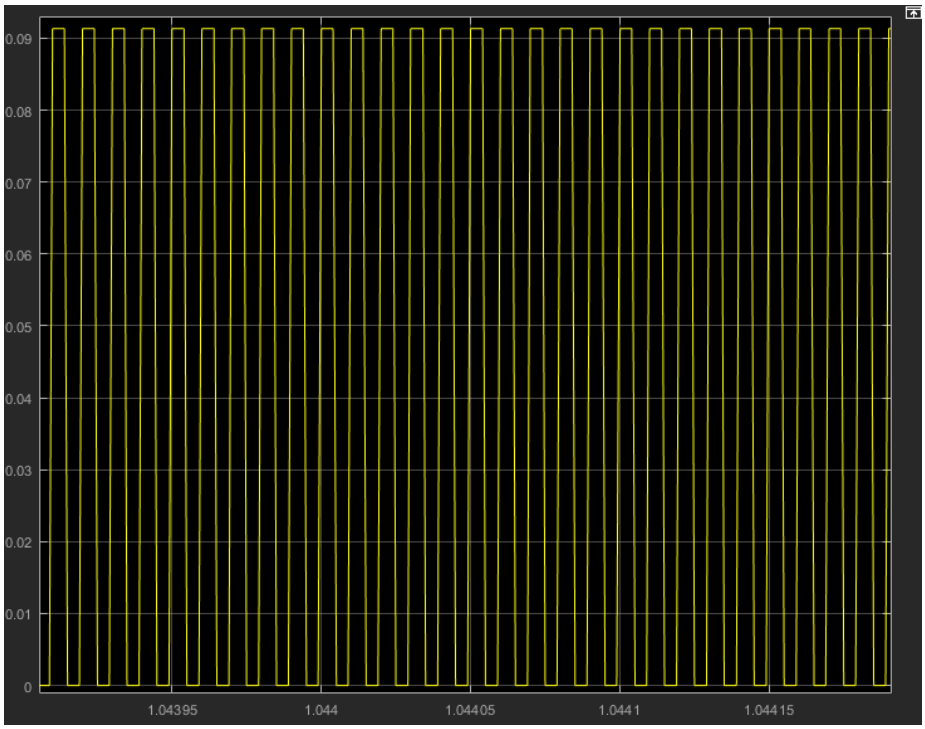
As we can see the final value is 1.826 μm, the overshoot is equal to 0.1 % and the Rise-Time is 2 ms. Corresponding capacitance variation ΔC is 15.5 fF:



The maximum value of the output signal is equal to 91 mV:



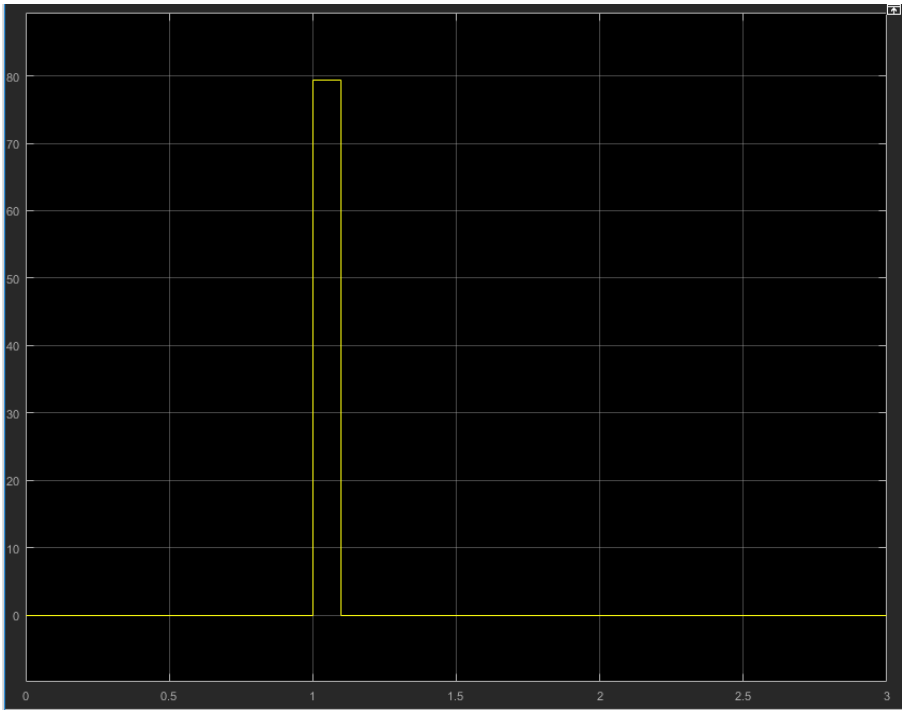
The output signal is a pulse waveform during the acceleration application:



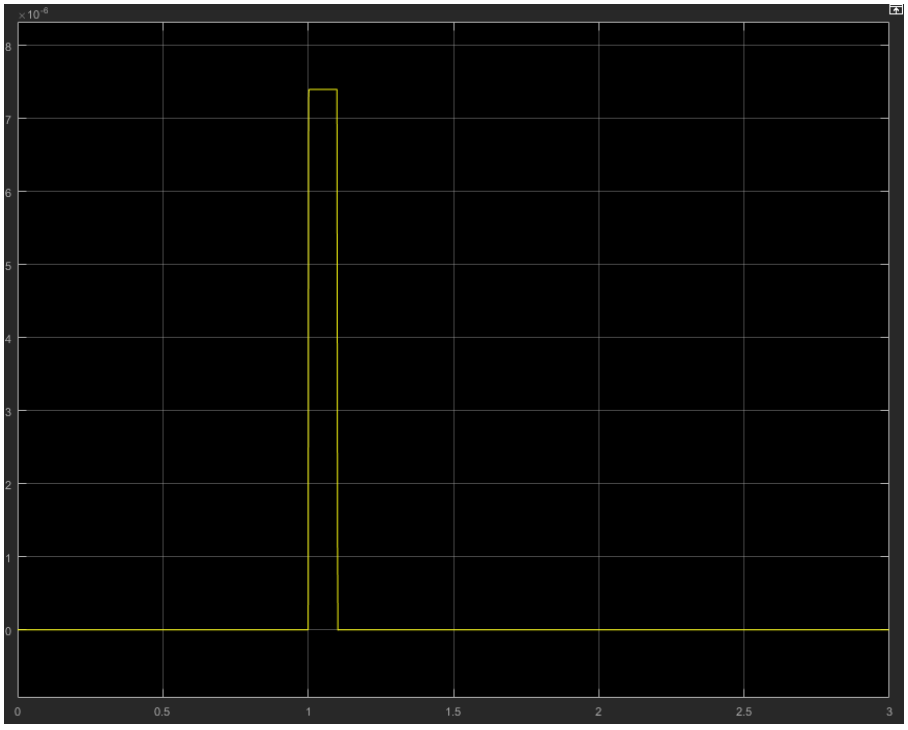
The outputs of **FrontComparator** and **RetroComparator** are zero. The resultant electrostatic force is a step with a maximum value of about $4 \cdot 10^{-9} N$.

3.5.4 Simulation: Positive Airbag Acceleration

The positive airbag acceleration is set to 8.1g and we have used a step input signal:

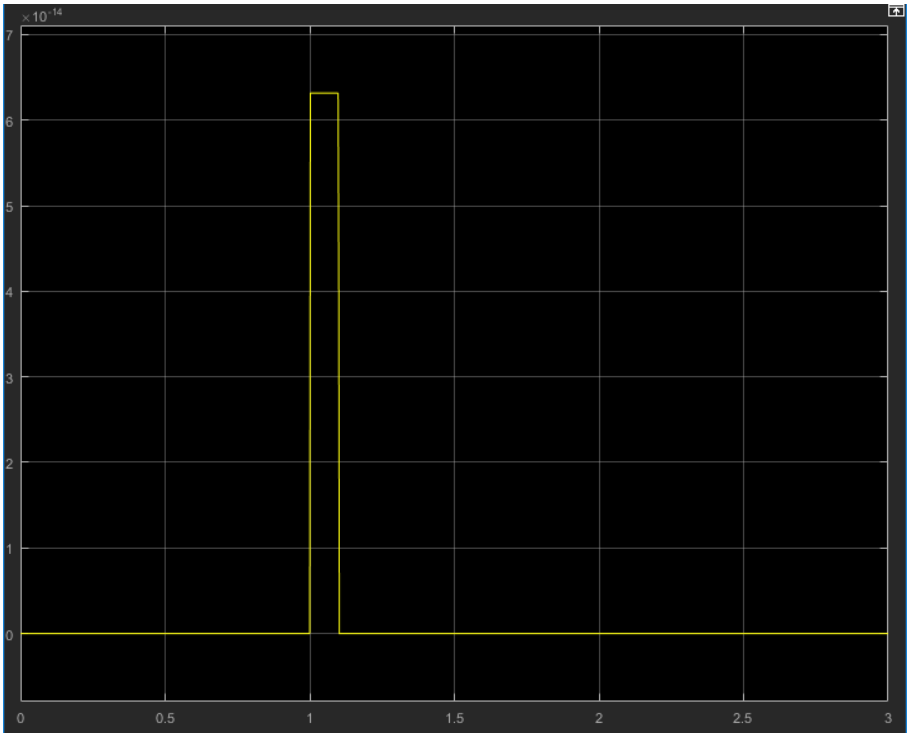


The resulting displacement g is:

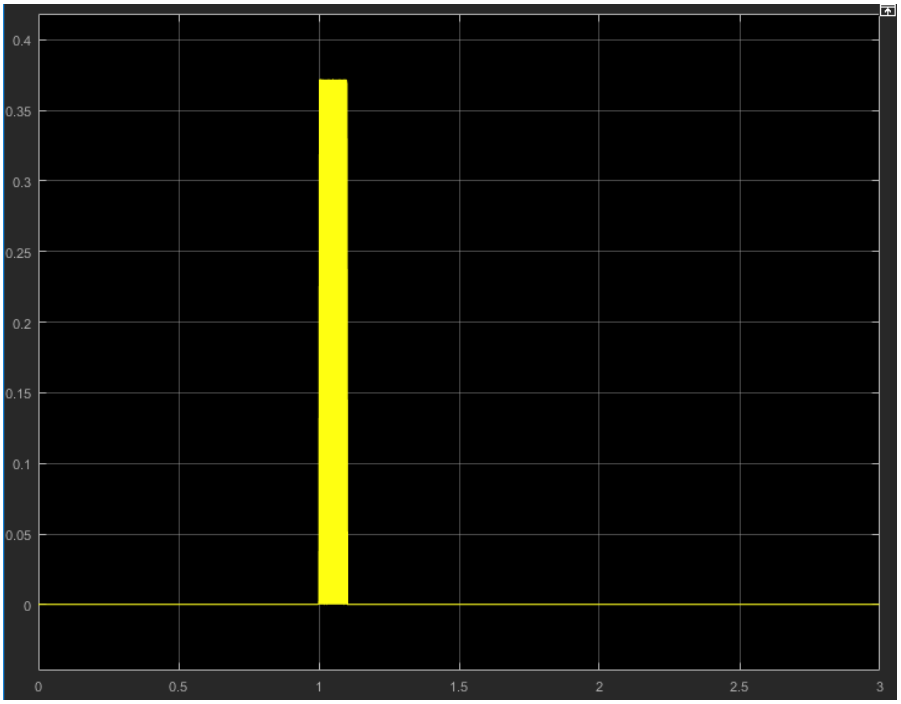


Using a zoomed image we found that g is equal to 7.397 μm. Also in this case the overshoot is equal to 0.1 % and the Rise-Time is 2 ms.

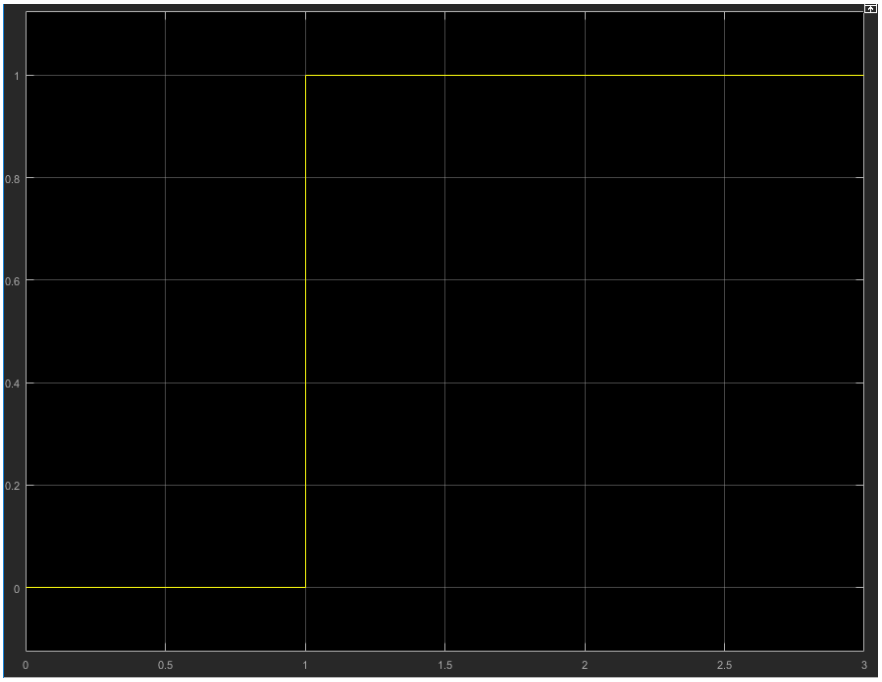
The corresponding capacitance variation ΔC is 63.2 fF:



The maximum value of the output signal V_O is equal to 0.372 V.



In this case the output of the **FrontComparator** is '1' because the positive airbag threshold has been exceeded. In fact the output of **FrontComparator** is:

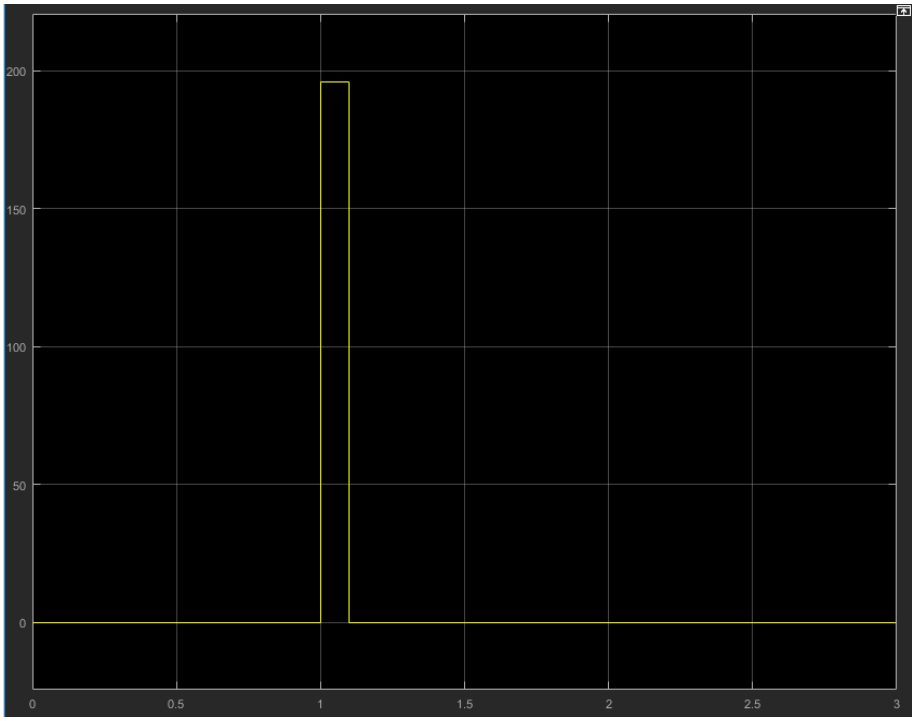


The behavior is exactly what we have expected. The output of **RetroComparator** is zero and the equivalent electrostatic force is a step like the previous case with a maximum value of about $16 \cdot 10^{-9} N$.

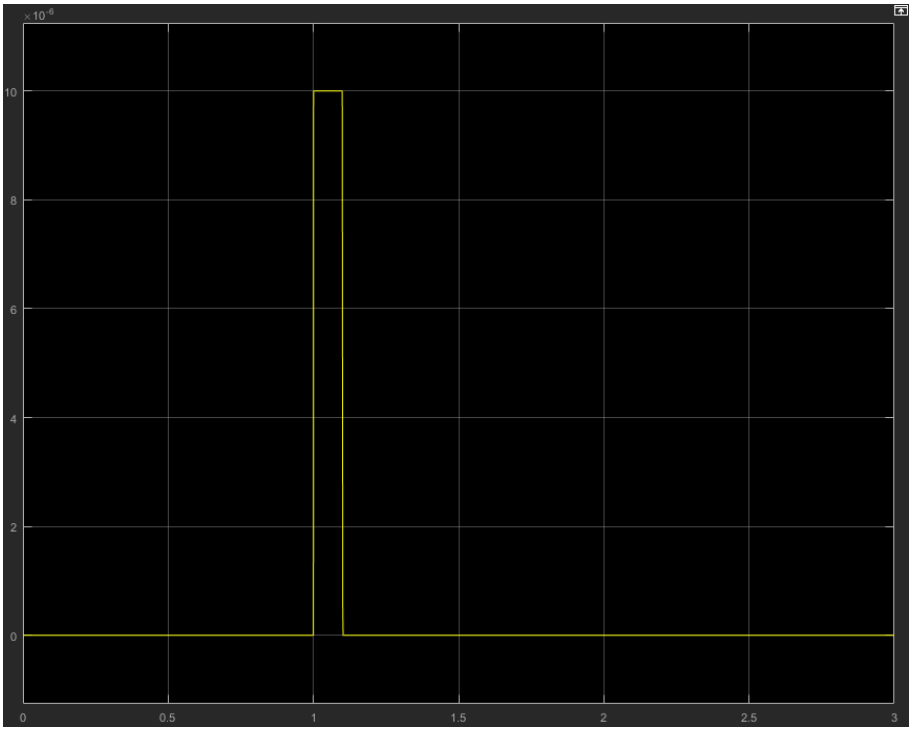
All simulated values respect the expected results.

3.5.5 Simulation: Maximum Positive Acceleration

The maximum positive acceleration is 10.96 g. We set the input acceleration to 20 g using a step input signal like the previous cases:

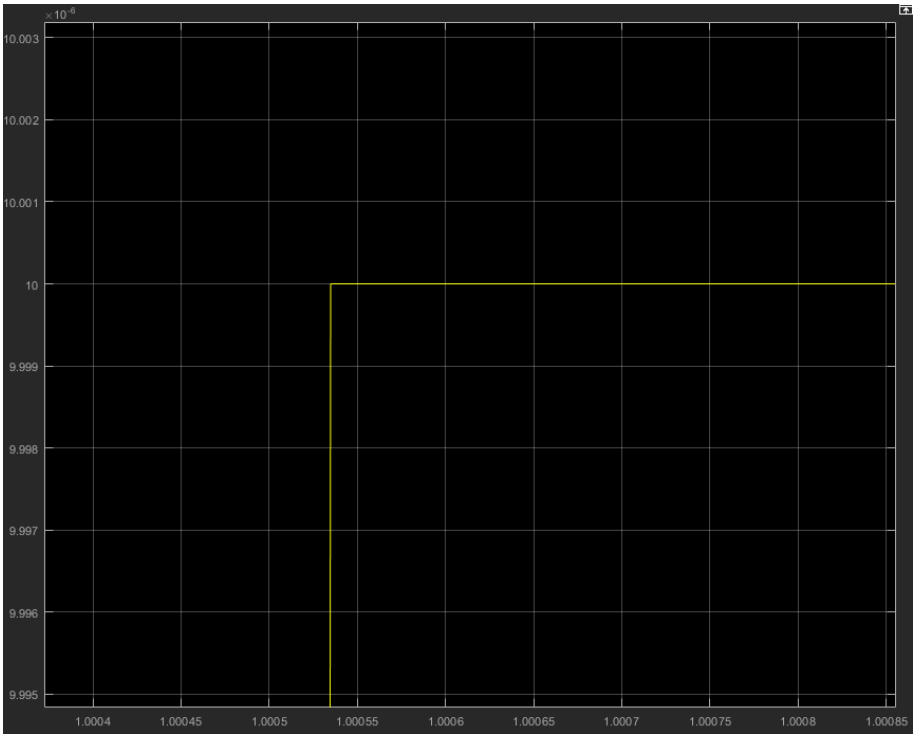


The resulting displacement g is:

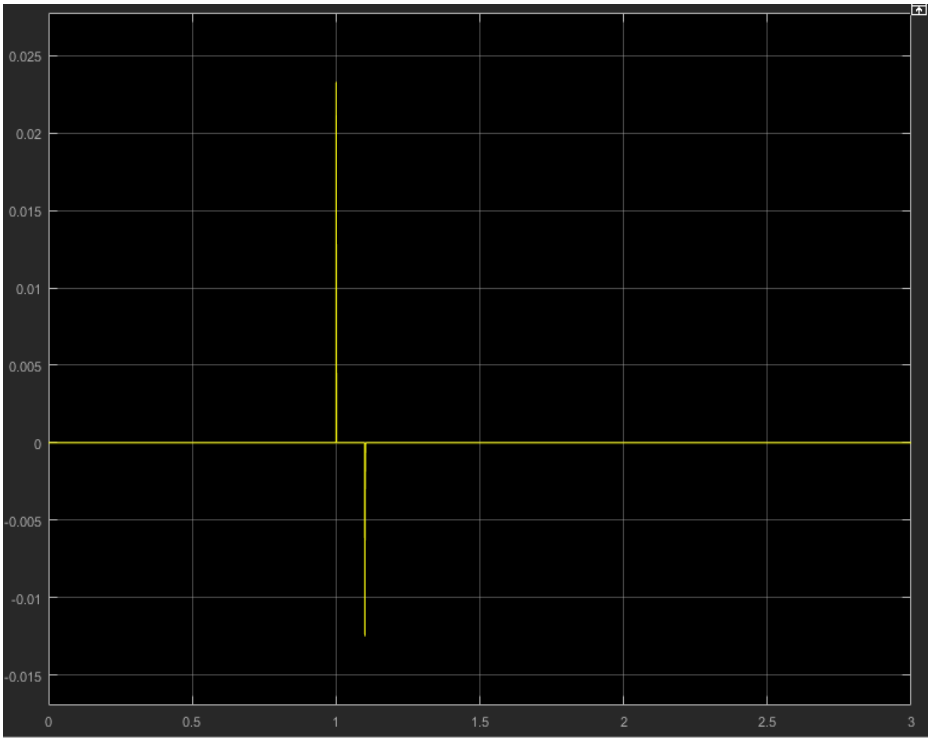


Using a zoomed image we found that g is equal to 10 μm and there is any overshoot because the saturation simulates the mechanical limit of our

accelerometer. The Rise-Time is 2 ms. Following image shows the zoomed displacement g:

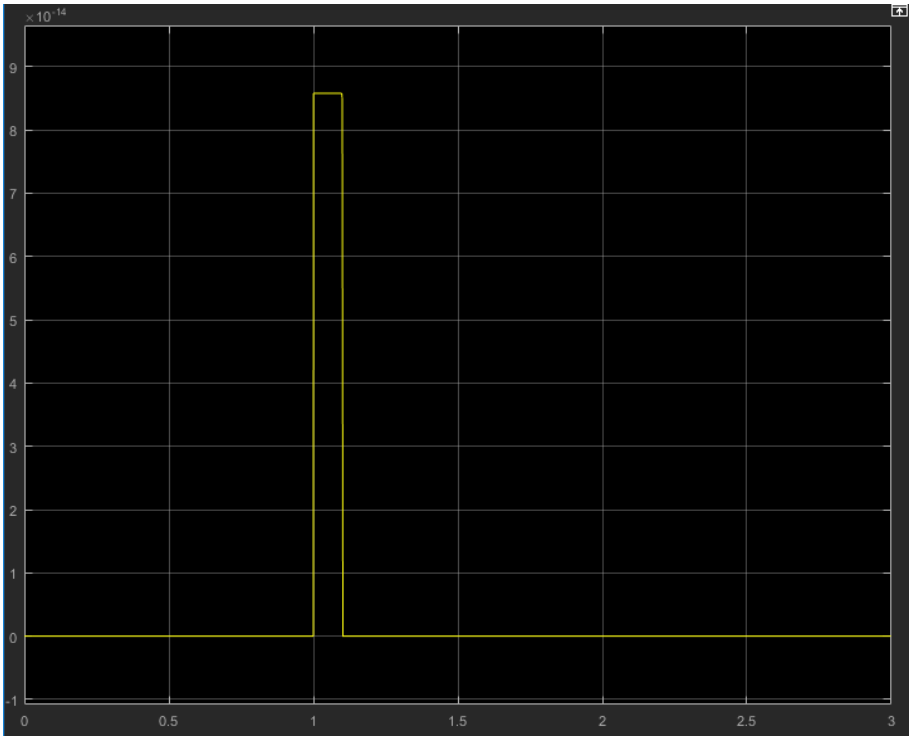


The following one shows the velocity behavior:

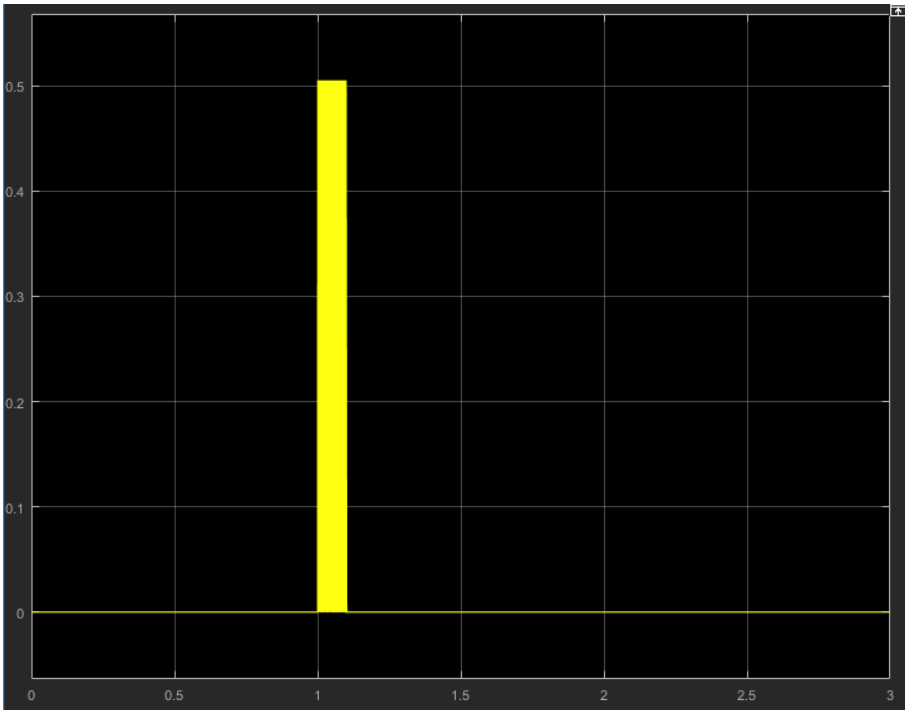


As we can see the velocity becomes zero at 1 second because the saturation occurs. When the input acceleration disappears the velocity becomes negative

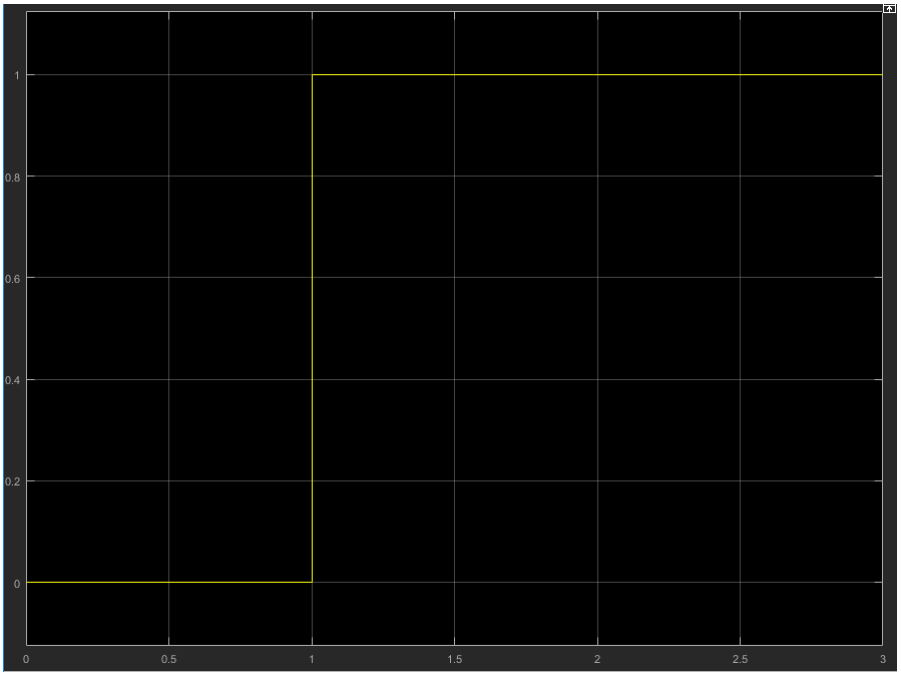
bringing the movable plate to the equilibrium position. The corresponding capacitance variation ΔC is 85.8 fF:



The maximum value of the output signal V_O is equal to 0.505 V:



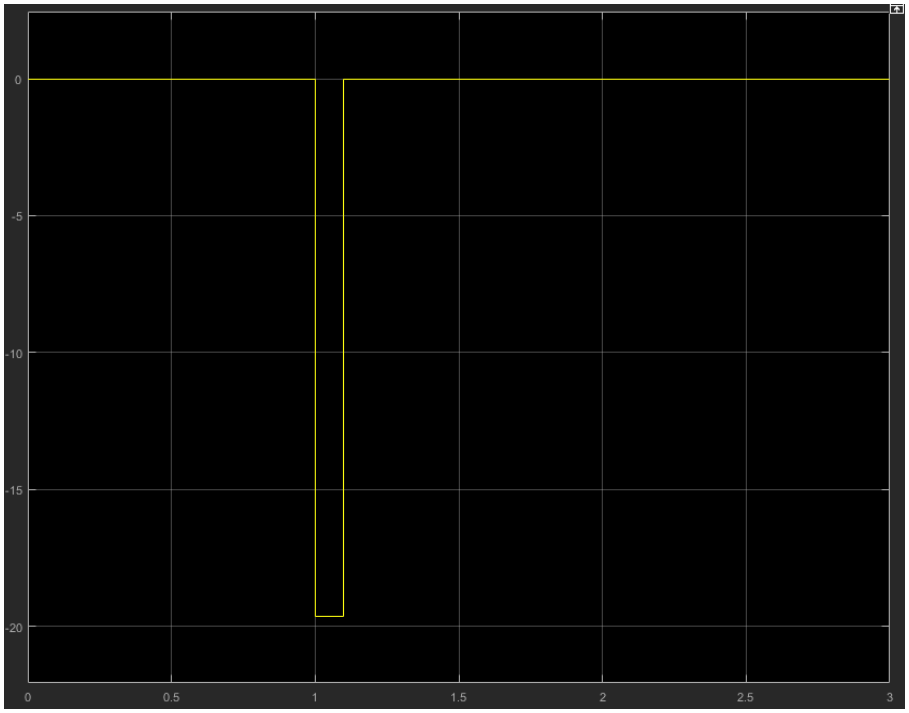
In this case the output of the **FrontComparator** is '1' logic because the positive airbag threshold was exceeded.



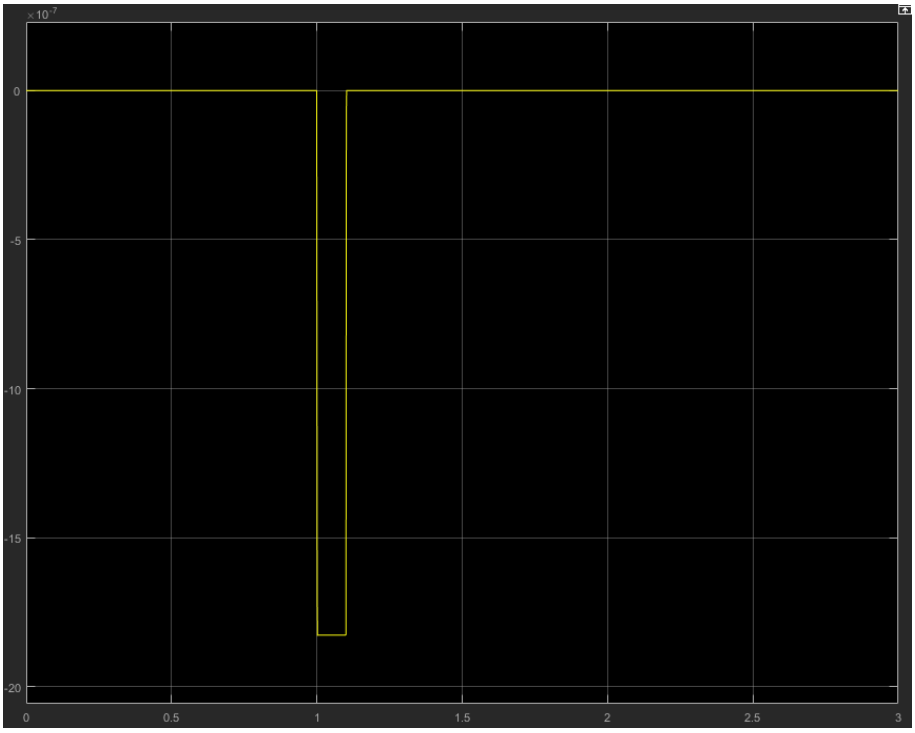
The behavior is exactly what we expected. The output of the **RetroComparator** is zero and the electrostatic force is a step like the previous cases with a maximum value of about $22 \cdot 10^{-9} N$ and it corresponds to an acceleration equal to $0.0788 \frac{m}{s^2}$, equal to the theoretical value. All simulated values respect the expected results.

3.5.6 Simulation: Negative Acceleration

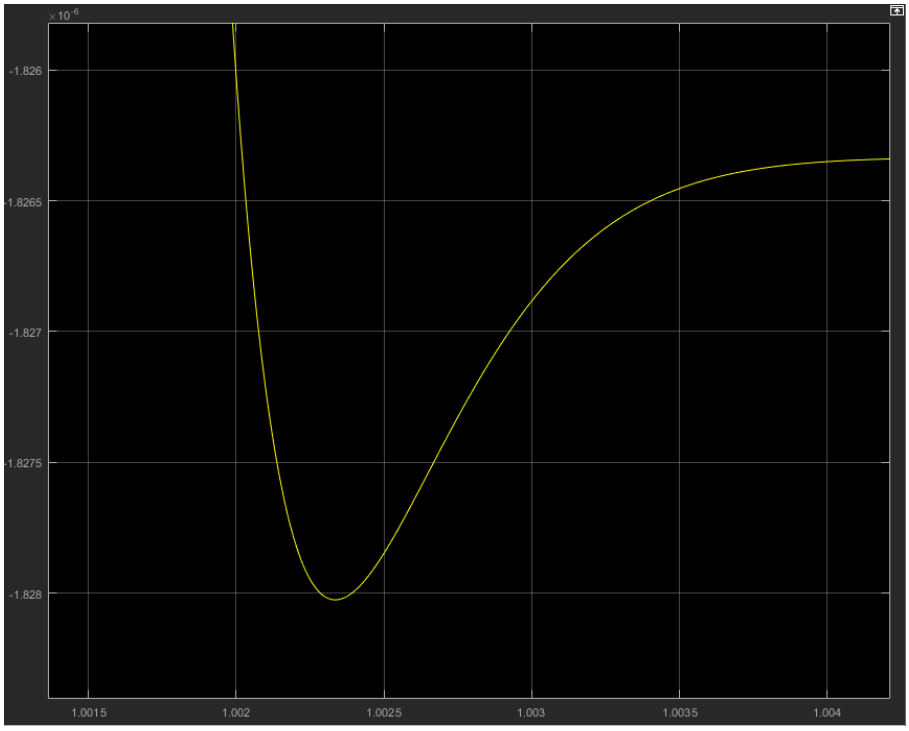
The negative Acceleration is set to −2 g and we have used a step input showed in the following figure:



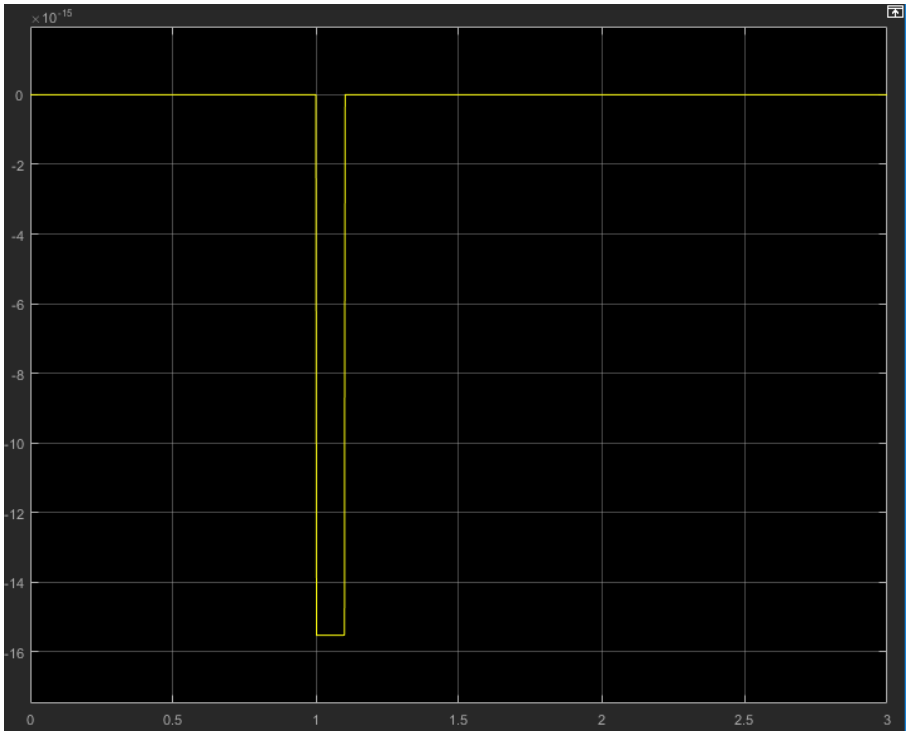
The resulting displacement g is:



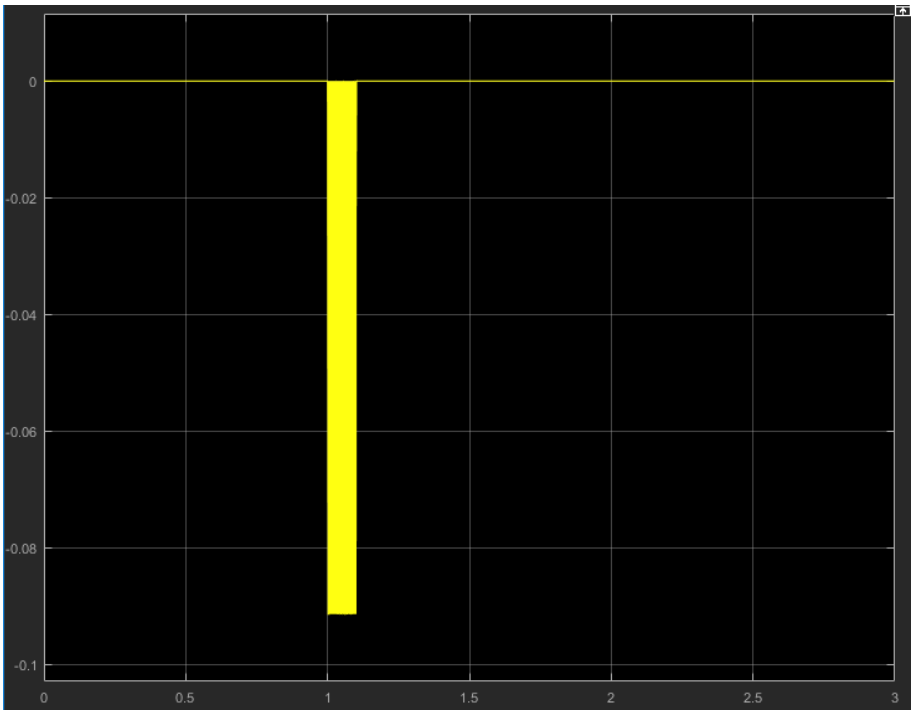
If we zoom the displacement behavior results:



As we can see the final value is −1.826 μm, the overshoot results equal to 0.1 % and the Rise-Time is 2 ms. Corresponding capacitance variation ΔC is −15.5 fF:



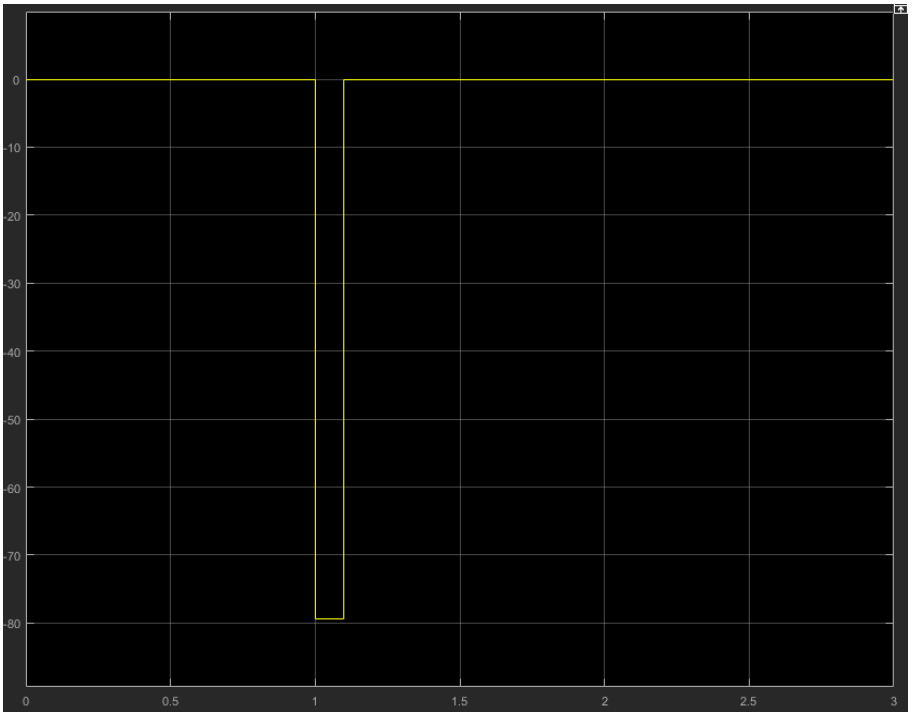
The absolute maximum value of the output signal is equal to 91 mV:



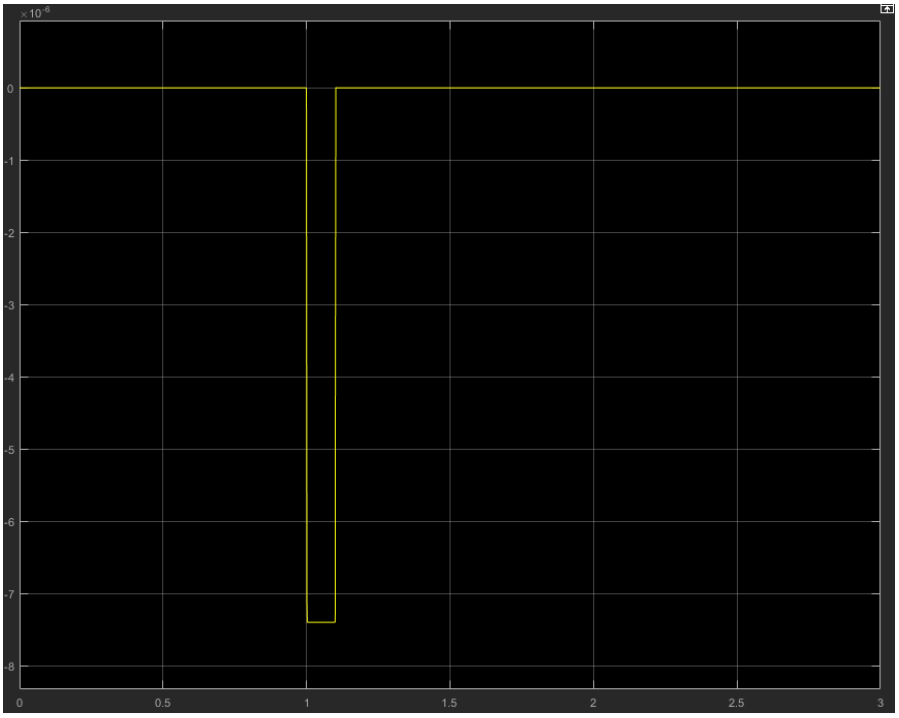
The outputs of **FrontComparator** and **RetroComparator** are zero. The electrostatic force is a step with an absolute maximum value of about $4 \cdot 10^{-9} N$. All simulated values respect the expected results.

3.5.7 Simulation: Negative Airbag Acceleration

The positive Airbag Acceleration is set to -8.1g and we have used a step input signal and it is showed in the following figure.

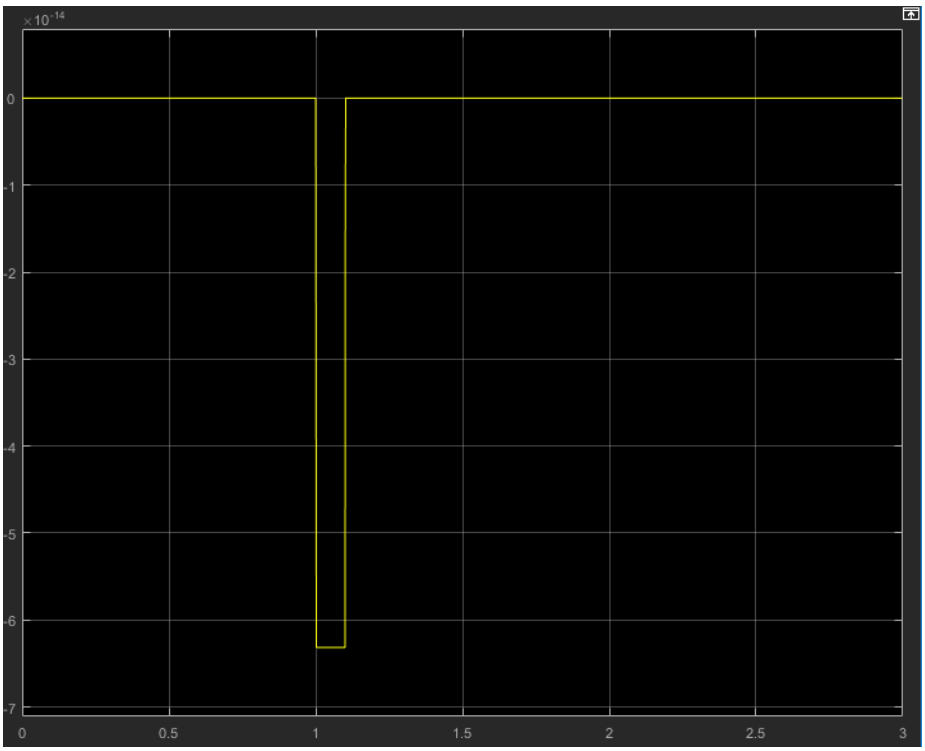


The resulting displacement g is:

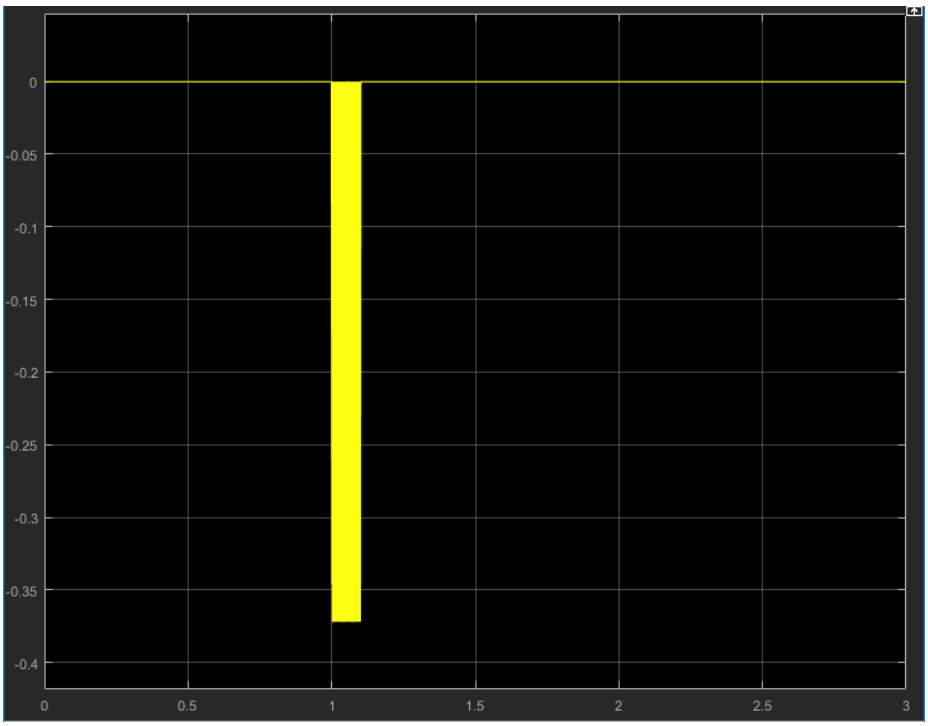


Using a zoomed image we found that g is equal to -7.397 μm. Also in this

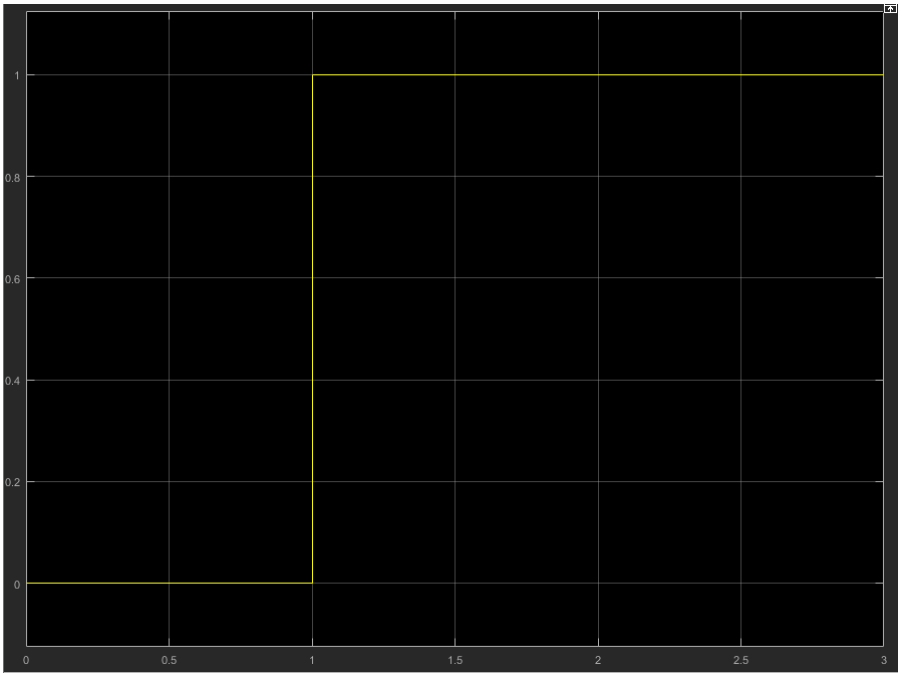
case Overshoot is equal to 0.1 % and the Rise-Time is 2 ms. Corresponding capacitance variation ΔC is -63.2 fF:



The absolute maximum value of the output signal V_O is equal to 0.372 V:



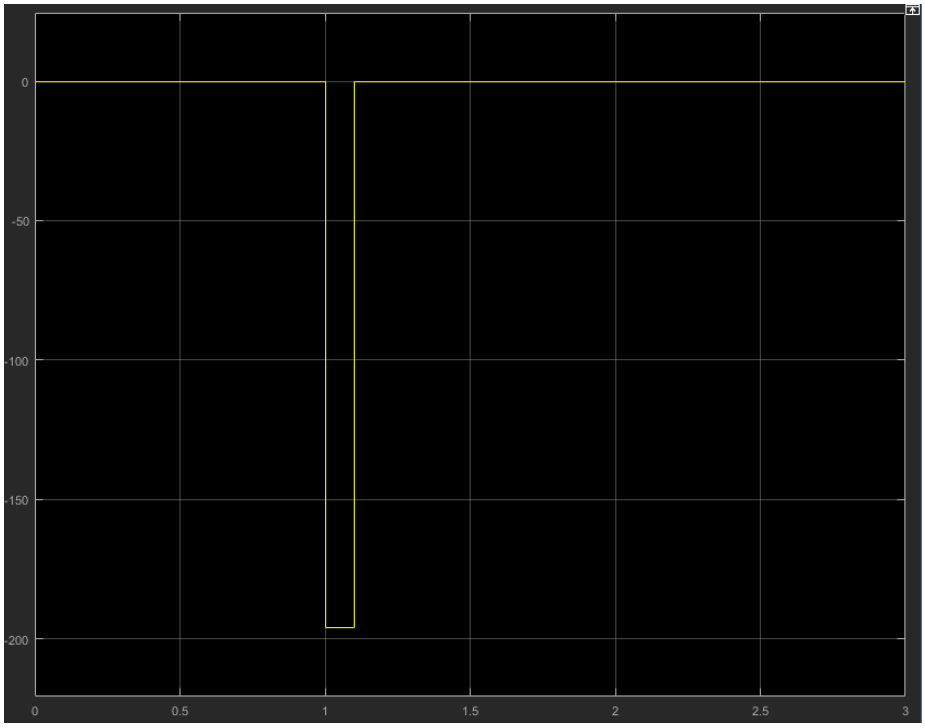
In this case the output of the **RetroComparator** is '1' because the negative airbag threshold was exceeded. The **RetroComparator** block output is:



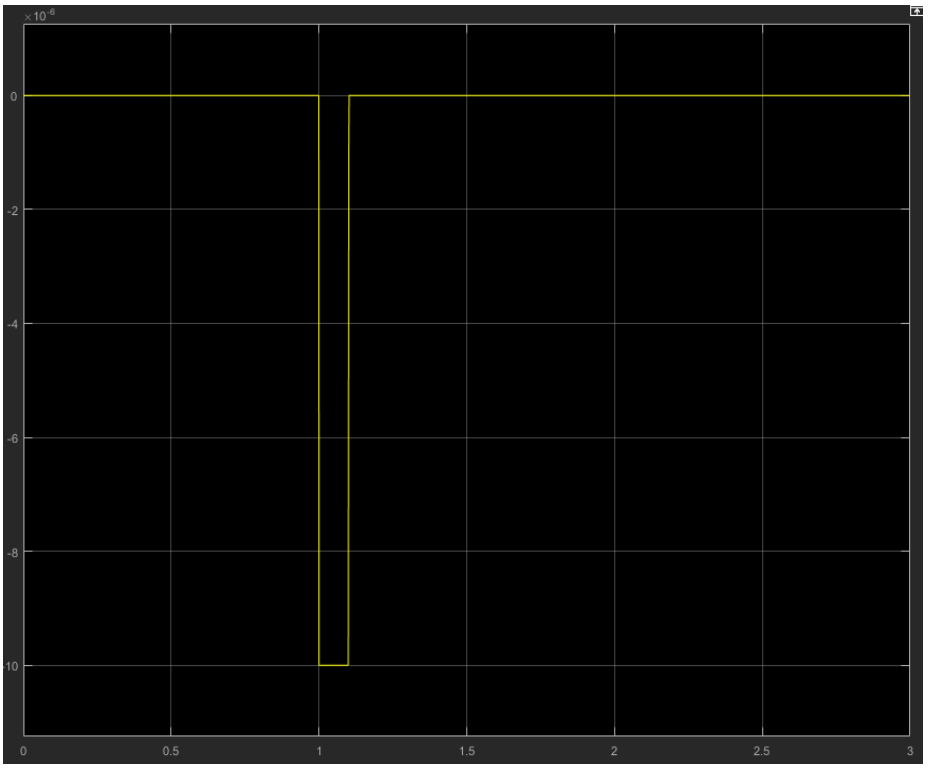
The behavior is exactly what we expected. The output of **FrontComparator** is zero and the electrostatic force is a step like the previous case with an absolute maximum value of about $16 \cdot 10^{-9} N$. All simulated values respect the expected results.

3.5.8 Simulation: Maximum Negative Acceleration

The maximum positive acceleration is -10.96 g. We set the input acceleration to -20 g using a step input signal like the previous cases:

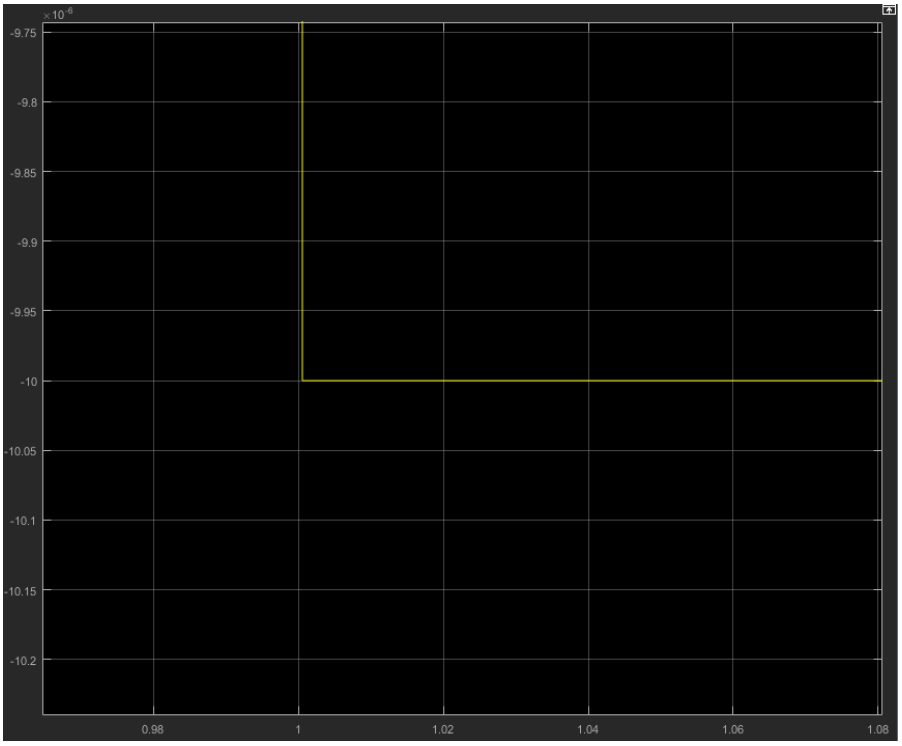


The resulting displacement g is:

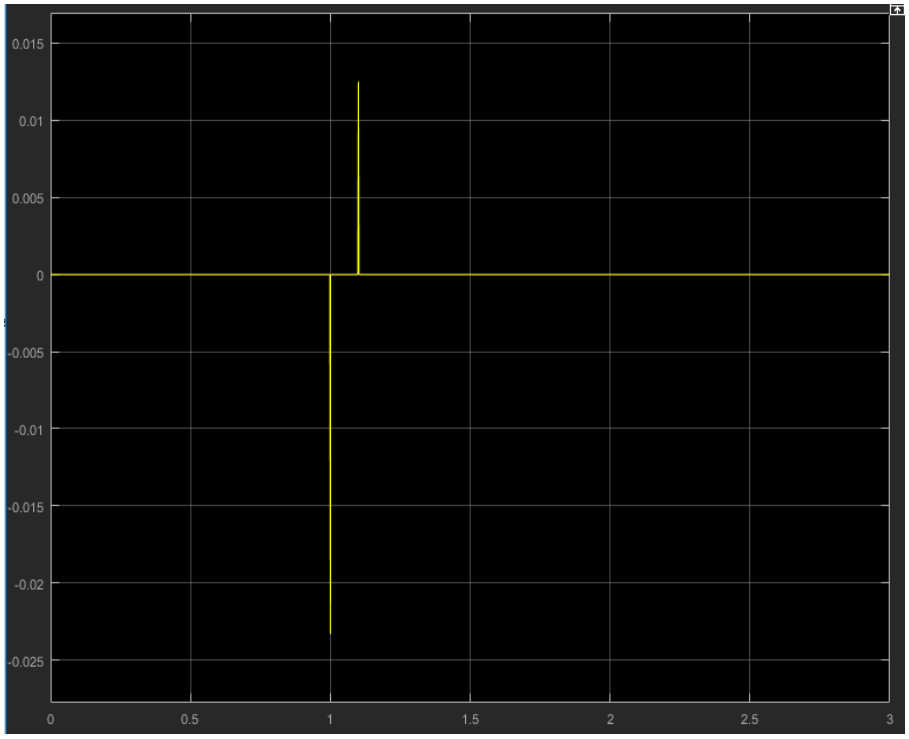


Using a zoomed image we found that g is equal to -10 μm and there is not

any overshoot because it is the mechanical limit of our accelerometer. The Rise-Time is 2 ms. Following image shows the zoomed displacement g:

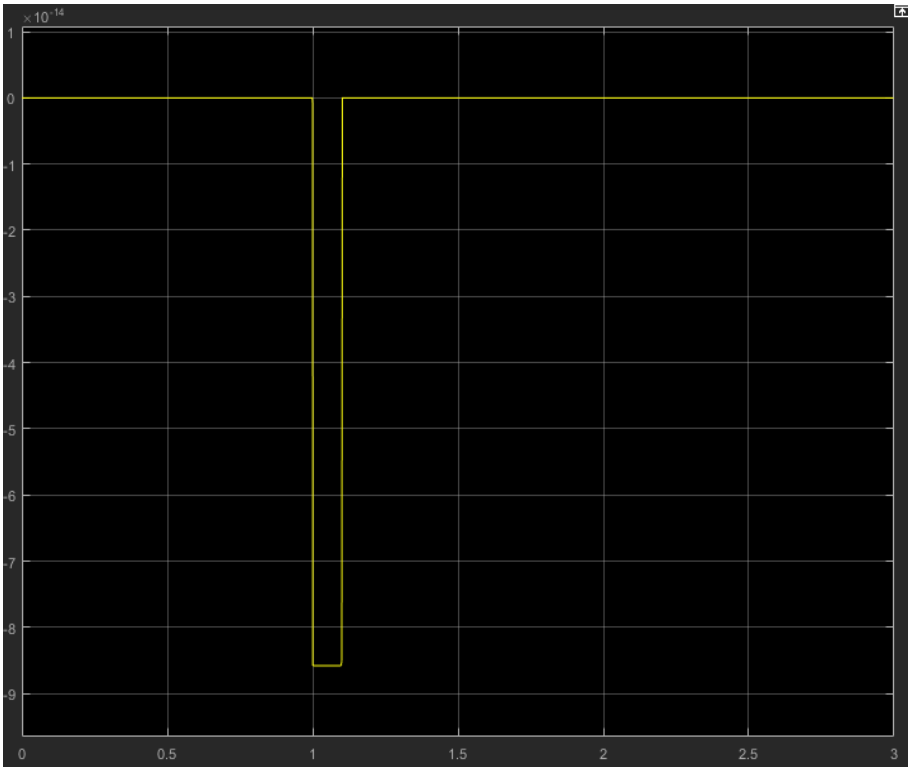


The image below shows the velocity behavior:

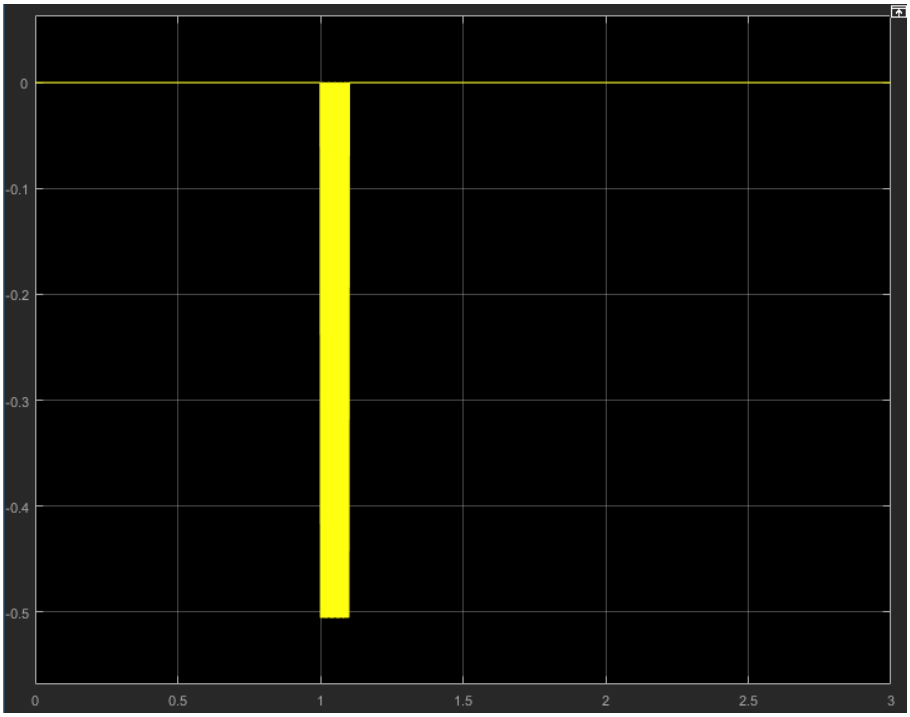


As we can see the velocity becomes zero at 1 second because the saturation occurs. When the input acceleration disappears the velocity becomes negative

bringing the movable plate to the equilibrium position. The corresponding capacitance variation ΔC is −85.8 fF:



The absolute maximum value of the output signal V_O is equal to 0.505 V:

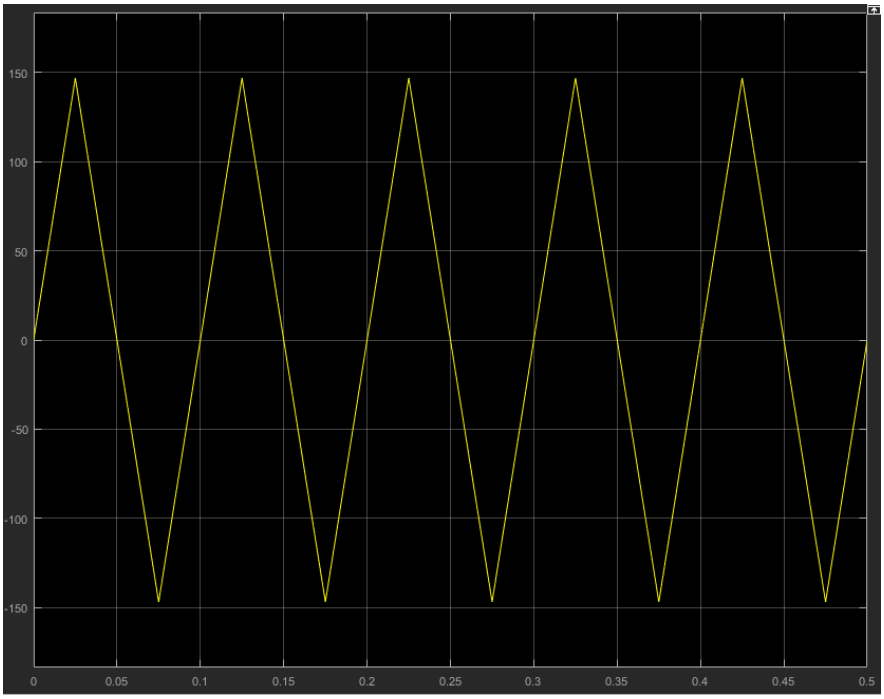


In this case the **RetroComparator** output is '1' because the negative airbag threshold was exceeded. The behavior is exactly what we expected. The output of **FrontComparator** is zero and the equivalent electrostatic force is a

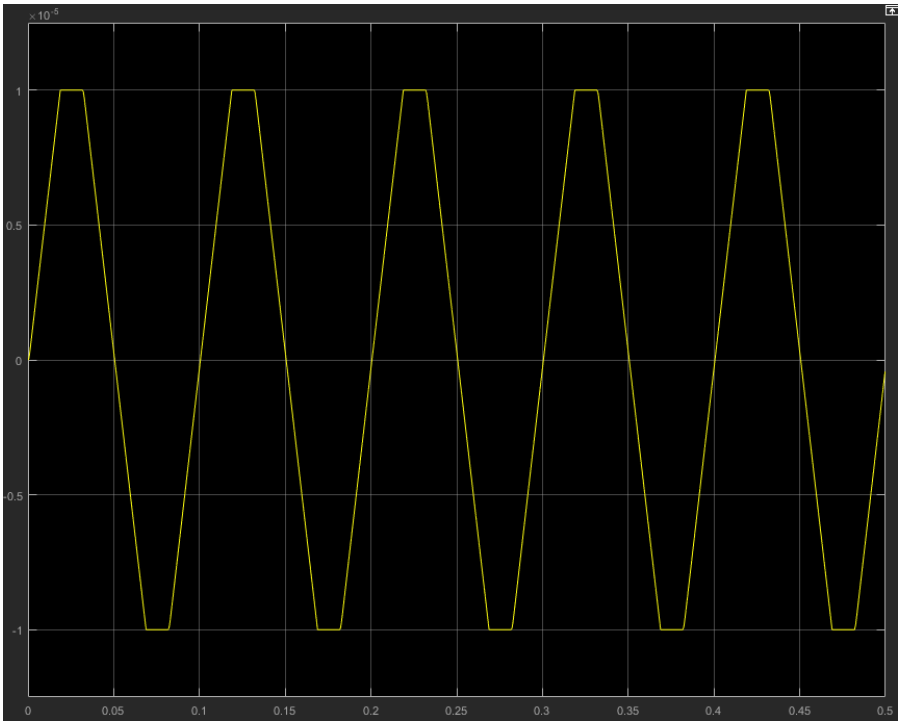
step like the previous cases with an absolute maximum value of about $22 \cdot 10^{-9}\ N$ and it corresponds to $0.0788\ \frac{m}{s^2}$, equal to the theoretical value. Also in this case all simulated values respect the expected results.

3.5.8 Simulation: Triangular Input Acceleration

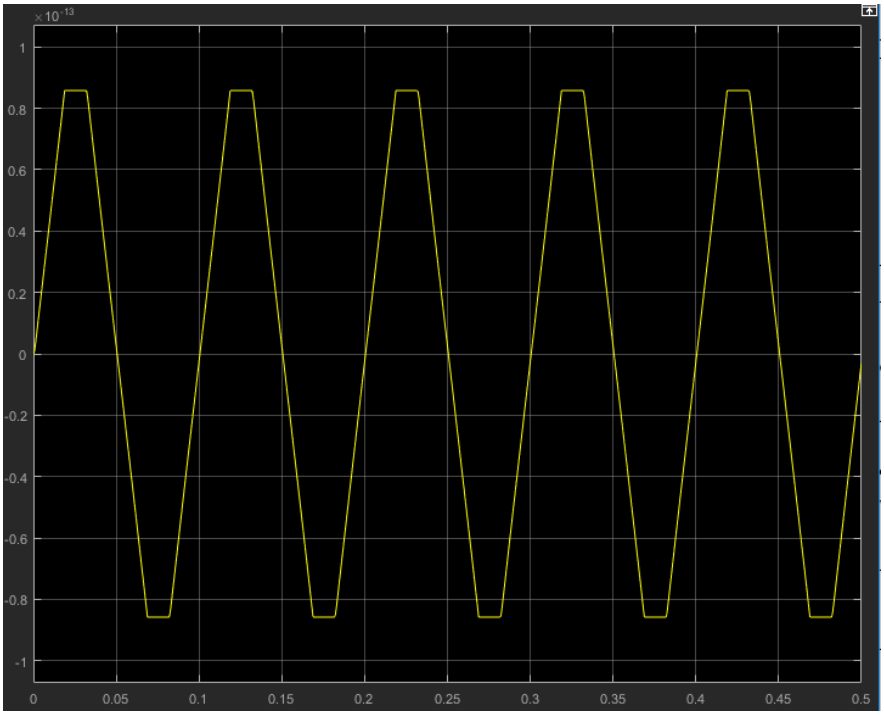
In order to observe the dynamic response of our system we have to consider an input ramp acceleration from −15 g to 15 g with a frequency of 10 Hz. This frequency corresponds to the period of a typical crash acceleration:



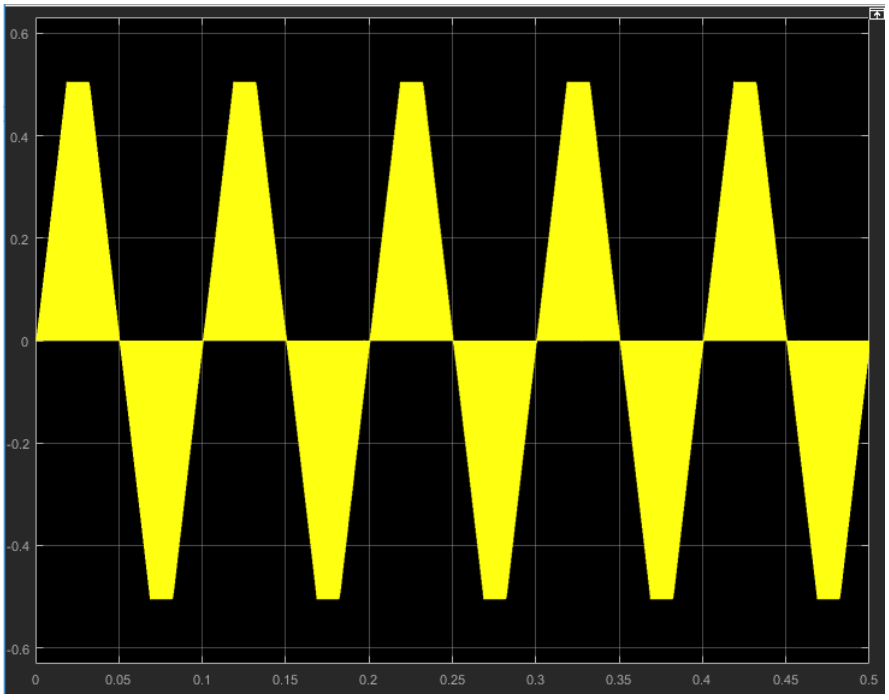
The resulting displacement is:



The corresponding capacitance variation is:



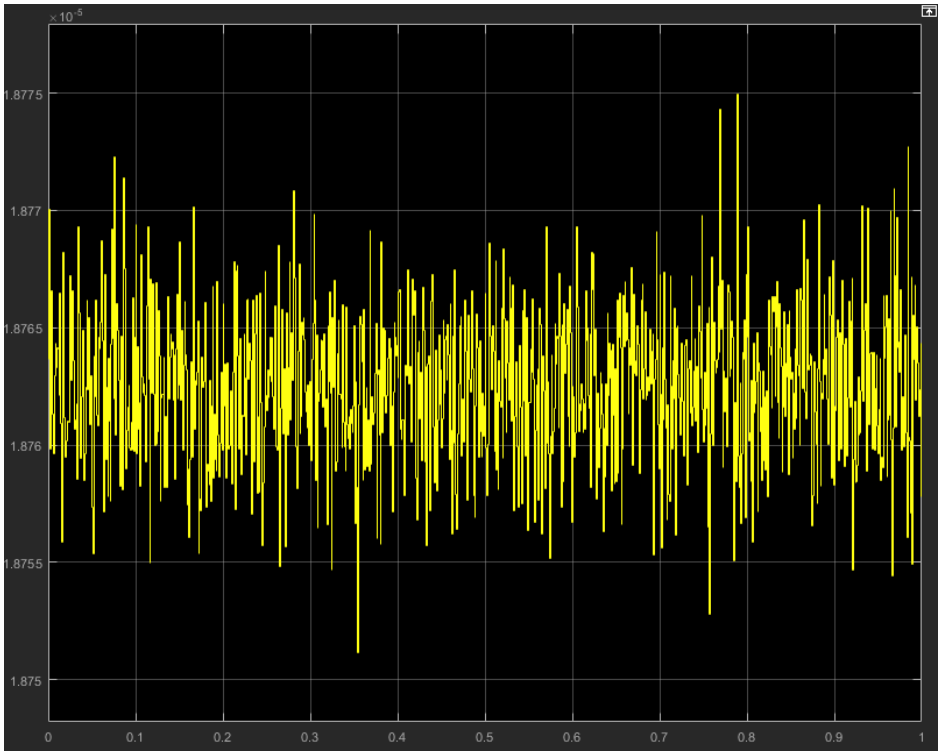
The output voltage is:



As we can see it varies between -0.5 and 0.5 V. It has the same shape of the input acceleration. The response of our system respects the expected results.

3.5.9 Simulation: Brownian Noise

In the following figure is showed the equivalent acceleration of Brownian noise used for all simulations.



The average value corresponds to the theoretical one.

3.6 Simulation: VHDL-AMS

3.6.1 VHDL-AMS Simulation Tools

For the VHDL-AMS simulation there are a lot of tools but the majority are designed for an industry application and for this reason they are not free. The most famous free programs are Smash and Simplorer. The first one is code oriented simulator. This means that for a simulation you need to create a specify file that describe the stimulus and the measures that you desire. The language used is similar to Spice.

Simplorer uses a graphical interface similar to Simulink and it allows a very simple simulation approach. After the VHDL-AMS description you can create a graphical component automatically that can be used inside the schematic with many other components of multiple domains. For this reason we have chosen Simplorer.

3.6.2 VHDL-AMS Modelling

The code of our accelerometer is showed below.

```
---------- VHDLAMS MODEL accelerometer ----------
LIBRARY ieee;
use ieee.electrical_systems.all;
use ieee.mechanical_systems.all;
use ieee.thermal_systems.all;
use ieee.math_real.all;
use ieee.std_logic_1164.all;

---------- ENTITY DECLARATION accelerometer ----------
ENTITY  accelerometer IS
GENERIC(m : MASS := 2.790588308929554e-07;
b0:  DAMPING := 0.001665767571543;
C0: CAPACITANCE:=8.401577831816385e-13;
u0: VISCOSITY:= 1.81e-5;
Ampiezza: VOLTAGE :=5.0;
k : STIFFNESS := 3.0;
A : REAL := 9.489019462182500e-06;
g0 : DISPLACEMENT := 100.0e-6;
g_min: DISPLACEMENT := 90.0e-6;
e0: REAL := 8.854e-12;
kb: REAL :=1.3806488e-23;
wn: REAL :=3.278783509396149e+03;
T0: TEMPERATURE := 298.0;
Q: REAL :=0.549280408932217;
ThreShold: REAL :=0.367);

PORT(Reset: In bit ;
OutFront: OUT bit:='0';
```

```
OutRetro: OUT bit:='0';
TERMINAL ThermalInput: THERMAL;
TERMINAL MechanicalOutput: TRANSLATIONAL;
TERMINAL InputAcceleration: TRANSLATIONAL;
TERMINAL IN1 : ELECTRICAL;
TERMINAL OUT1 : ELECTRICAL);
END ENTITY accelerometer;

---------- ARCHITECTURE DECLARATION simple ----------

ARCHITECTURE simple OF accelerometer IS

QUANTITY Vin ACROSS IN1 TO Electrical_Ref;
QUANTITY Vout ACROSS Iout THROUGH OUT1 to electrical_ref;
QUANTITY g ACROSS Fext THROUGH InputAcceleration;
QUANTITY gout ACROSS FRis THROUGH MechanicalOutput;
QUANTITY T ACROSS ThermalInput;
QUANTITY Fel: FORCE:=0.0;
QUANTITY deltaC: CAPACITANCE:=0.0;
QUANTITY BrownianNoise: ACCELERATION:=0.0;
SIGNAL clockP,clockN: std_logic:='0';

BEGIN

if g>(g0-g_min) USE
g==(g0-g_min);
elsif g<(-g0+g_min) USE
g==(-g0+g_min);
else
g'dot'dot==BrownianNoise*m + Fext/m - k*g/m -1.0/2.0*u0*A**2.0*(1.0/(g0-
g)**3.0+1.0/(g0+g)**3.0)*g'dot/m+Fel/m;
end use;

Fel==(Ampiezza**2.0*e0*A)/4.0*( 1.0/(g0-g)**2.0-1.0/(g0+g)**2.0);
BrownianNoise==sqrt(4.0*kb*(T0+T)*wn/Q/m);
gout==-(-g);
deltaC==e0*A*g/(g0**2.0-g**2.0);
Vout==Vin*deltaC/C0;

process (Vout'ABOVE(Threshold)) is
BEGIN
if Vout'ABOVE(Threshold) then
clockP<='1';
else
clockP<='0';
end if;
end process;



process (Vout'ABOVE(-Threshold)) is
```

```
BEGIN
if NOT Vout'ABOVE(-Threshold) then
clockN<='1';
else
clockN<='0';
end if;
end process;

Process(clockP,Reset)is
BEGIN
if(Reset='0')then
OutFront<='0';
elsif(clockP'event AND ClockP='1')then
OutFront<='1';
end if;
end process;

Process(clockN,Reset)is
BEGIN
if(Reset='0')then
OutRetro<='0';
elsif(clockN'event AND ClockN='1')then
OutRetro<='1';
end if;
end process;
END ARCHITECTURE simple;
```

Observing the model, we note that there are the two sections: Entity and Architecture. The Entity includes the GENERIC statement for the definition of all parameters used in our system. Moreover it includes the PORT section that describes both digital ports and terminals:

- *Reset*: digital input for the reset of two D Flip-Flops;
- *OutFront*: digital output of the positive threshold D Flip-Flop;
- *OutRetro*: digital output of the negative threshold D Flip-Flop;
- *ThermalInput*: thermal input for the external temperature;
- *MechanicalOutput*: translational output for the displacement g;
- *InputAcceleration*: translational input for the external acceleration;
- *IN1*: electrical input for the bias generator V_1;
- *OUT1*: electrical output for the output signal V_O;

Inside the Architecture section we found the definition of all quantities in order to assign them to the corresponding terminal. We have also defined internal

quantities and digital signals. After the BEGIN statement we found both continuous and discrete descriptions.

Continuous-Time Section

The description is the following:

```
if g>(g0-g_min) USE
g==(g0-g_min);
elsif g<(-g0+g_min) USE
g==(-g0+g_min);
else
g'dot'dot==BrownianNoise*m + Fext/m - k*g/m -1.0/2.0*u0*A**2.0*(1.0/(g0-
g)**3.0+1.0/(g0+g)**3.0)*g'dot/m+Fel/m;
end use;

Fel==(Ampiezza**2.0*e0*A)/4.0*( 1.0/(g0-g)**2.0-1.0/(g0+g)**2.0);
BrownianNoise==sqrt(4.0*kb*(T0+T)*wn/Q/m);
gout==-(-g);
deltaC==e0*A*g/(g0**2.0-g**2.0);
Vout==Vin*deltaC/C0;
```

As we can see there is the "If-Elsif-Else" statement for the saturation modeling for both negative and positive limit. If no saturations occur, the system uses the complete differential equation in order to calculate g variation:

$$m \cdot g'' = m \cdot (a_{ext} + a_{BW}) - k \cdot g - \frac{1}{2} \cdot u \cdot A^2 \left(\frac{1}{(g_0 - g)^3} + \frac{1}{(g_0 + g)^3} \right) g' + \frac{\varepsilon_0 \cdot A \cdot V_P^2}{4} \left[\frac{1}{(g_0 - g)^2} - \frac{1}{(g_0 + g)^2} \right]$$

Here the saturation description is simpler than Simulink. Since we use a differential equation modelling we do not have to set to zero the acceleration or the velocity because this happen automatically inside the equation. After the "If-Elsif-Else" statement, we find the Brownian noise, the ΔC and V_O formulas.

Discrete-Time Section

The description is showed below:

```
FFA:Process(clockP,Reset)is
BEGIN
if(Reset='0')then
OutFront<='0';
elsif(clockP'event AND ClockP='1')then
OutFront<='1';
end if;
end process;

FFB:Process(clockN,Reset)is
BEGIN
if(Reset='0')then
OutRetro<='0';
elsif(clockN'event AND ClockN='1')then
OutRetro<='1';
end if;
end process;
```

It includes two D Flip-Flops with two different clocks: ClockN and ClockP. These are the two outputs of the comparators. The first one is related to **RetroComparator** and the second one to **FrontComparator**. They are used in the same way described inside the Simulink model.

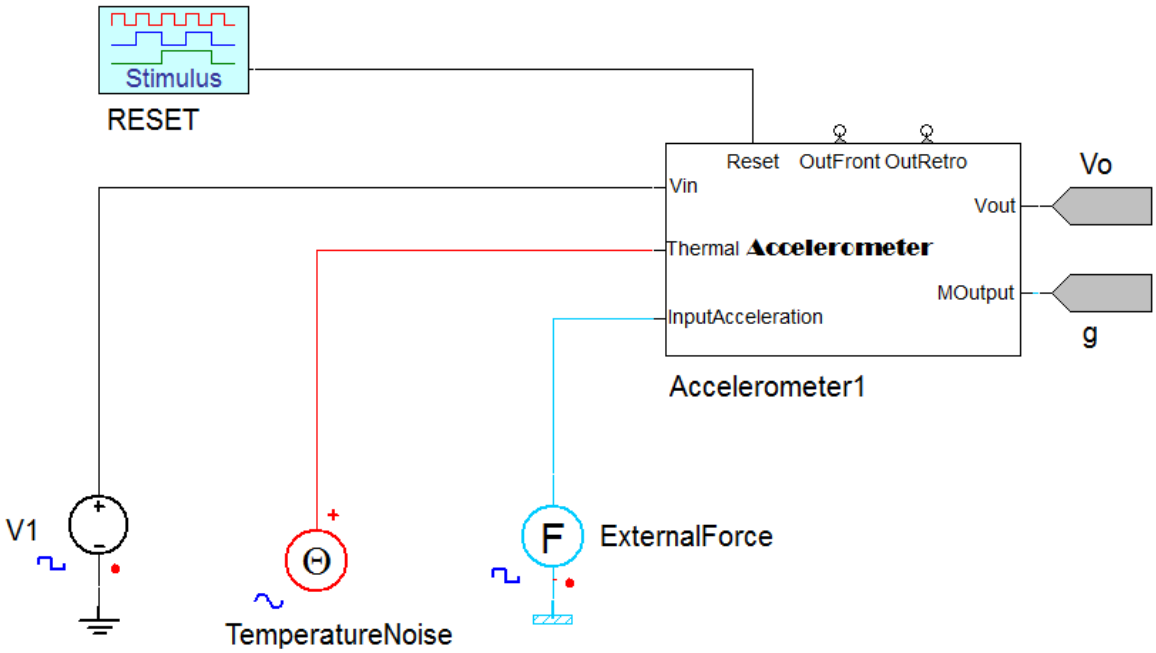
Continuous-Discrete Interface

The description is showed below:

```
process (Vout'ABOVE(Threshold)) is
BEGIN
if Vout'ABOVE(Threshold) then
clockP<='1';
else
clockP<='0';
end if;
end process;
process (Vout'ABOVE(-Threshold)) is
BEGIN
if NOT Vout'ABOVE(-Threshold) then
clockN<='1';
else
clockN<='0';
end if;
end process;
```

Two comparators are the connection between the continuous and discrete section. As described in the paragraph 2.2 we use two process, one for each comparator. The first one is the **FrontComparator** and it asserts the clockP signal. The second one is the **RetroComparator** and it asserts the clockN signal.

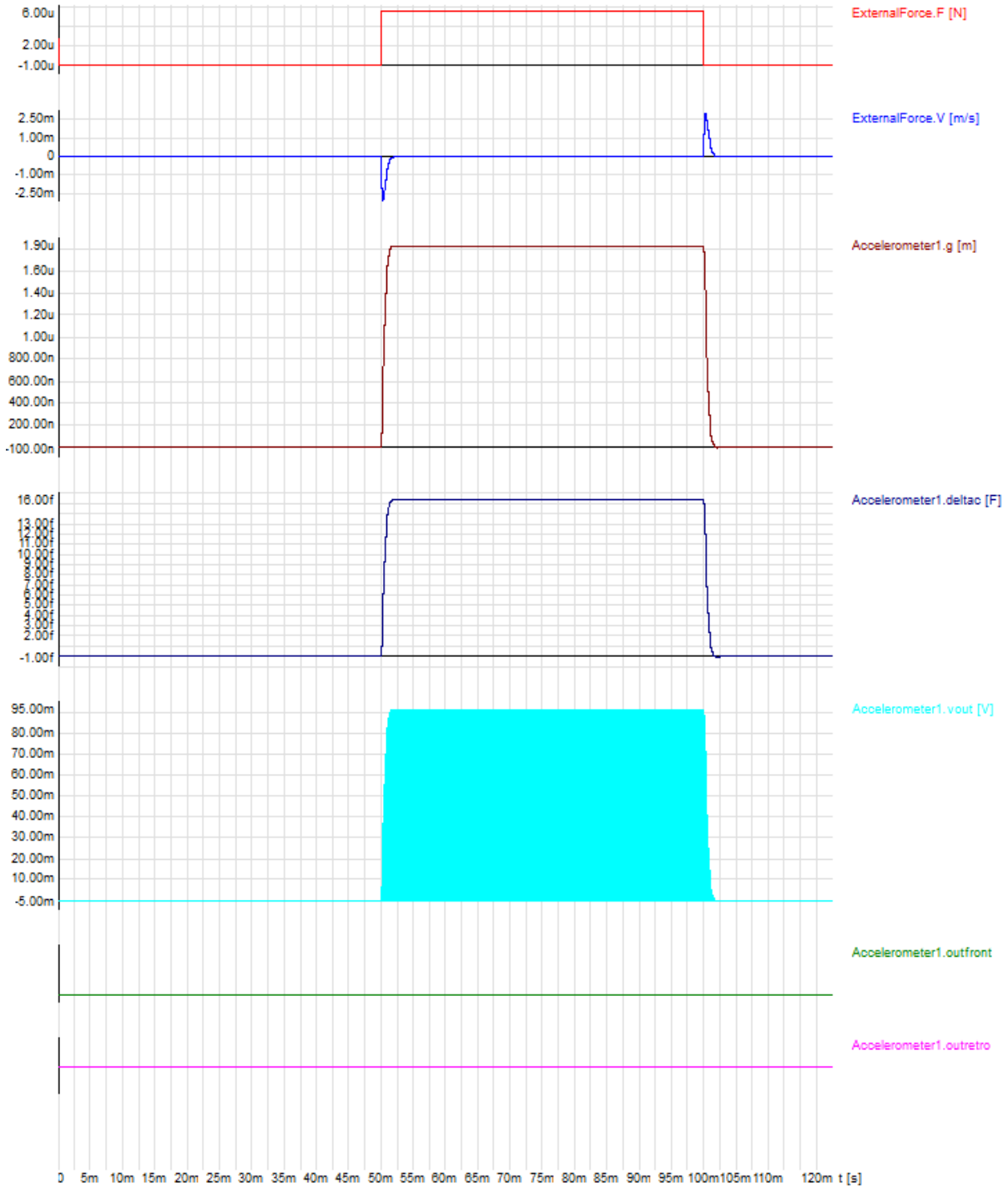
In the following paragraphs are showed the results of the simulations done with VHDL-AMS. They are comparable with respect to the Simulink model. Here the Brownian noise is simulated with a sinusoidal temperature noise over the average value of 298 K. The simulation schematic is showed below:



The accelerometer symbol was created through Simplorer thanks to the Symbol Editor tool included internally. It allowed us to define the shape and the pin disposition. The quality of simulation is more limited than Simulink because the graphical user interface is simpler with less functionalities. This limits the evaluation of the results. The pulse duration input for the acceleration is set to 100 ms, like in Simulink. The finally simulation use a triangular input acceleration in order to evaluate the dynamic response when g varies from -10 µm to 10 µm.

3.6.3 Simulation VHDL-AMS: Positive Acceleration

As the Simulink simulation we have used a positive step acceleration of 2 g. The figure below show the main signals obtained from the simulation.

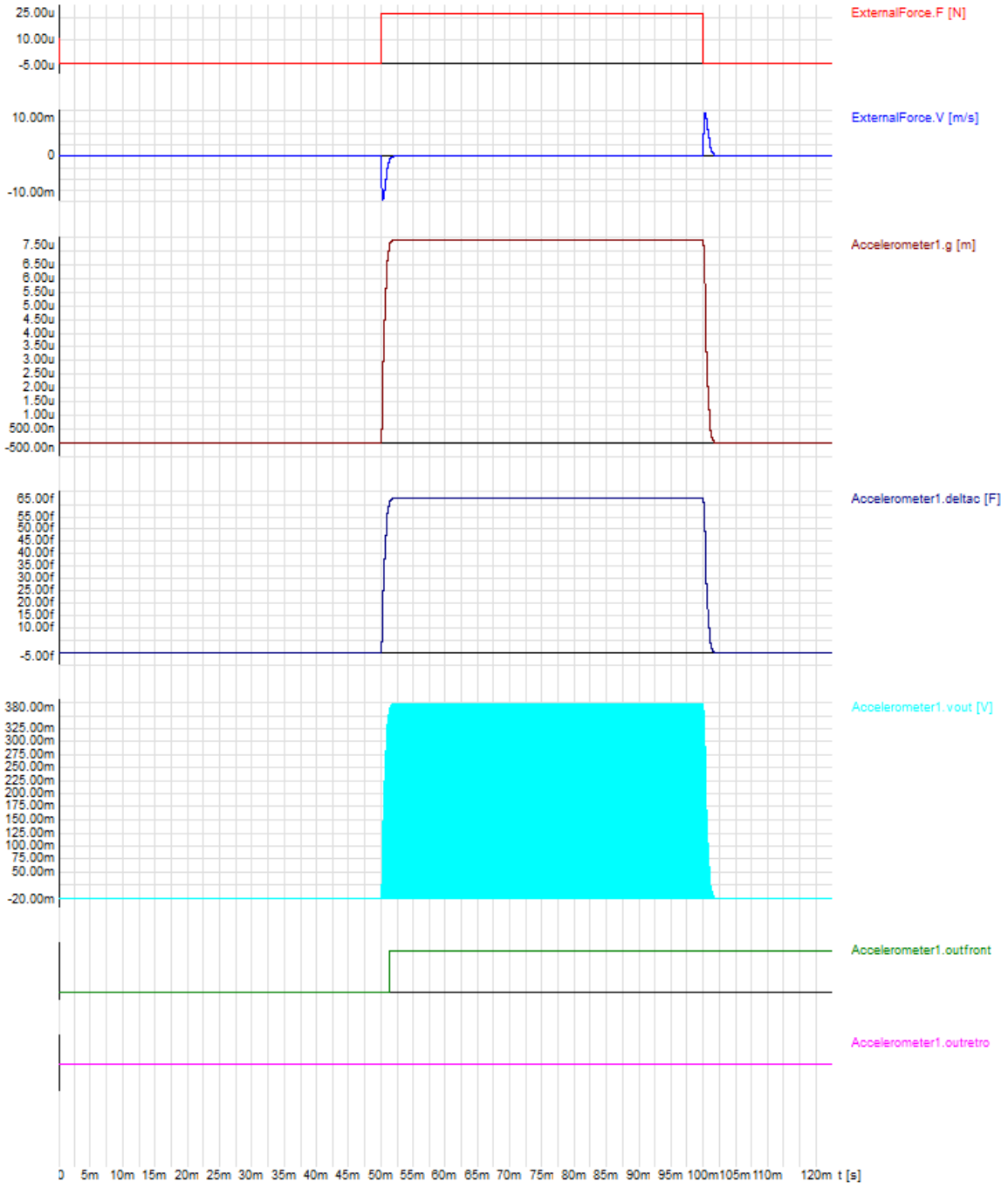


We can observe that the displacement g is about 1.9 µm and the corresponding capacitance variation ΔC is 15.5 fF. The maximum output voltage V_O is equal to about 95 mV. The comparators output is '0'. The

electrostatic force is a step with a maximum value of about $4 \cdot 10^{-9}N$. The Rise-Time is 2 ms and all simulated values respect the expected results.

3.6.4 Simulation VHDL-AMS: Positive Airbag Threshold Acceleration

In this case we have tested the positive airbag threshold using a step of 8.1 g. The figure below shows the results:

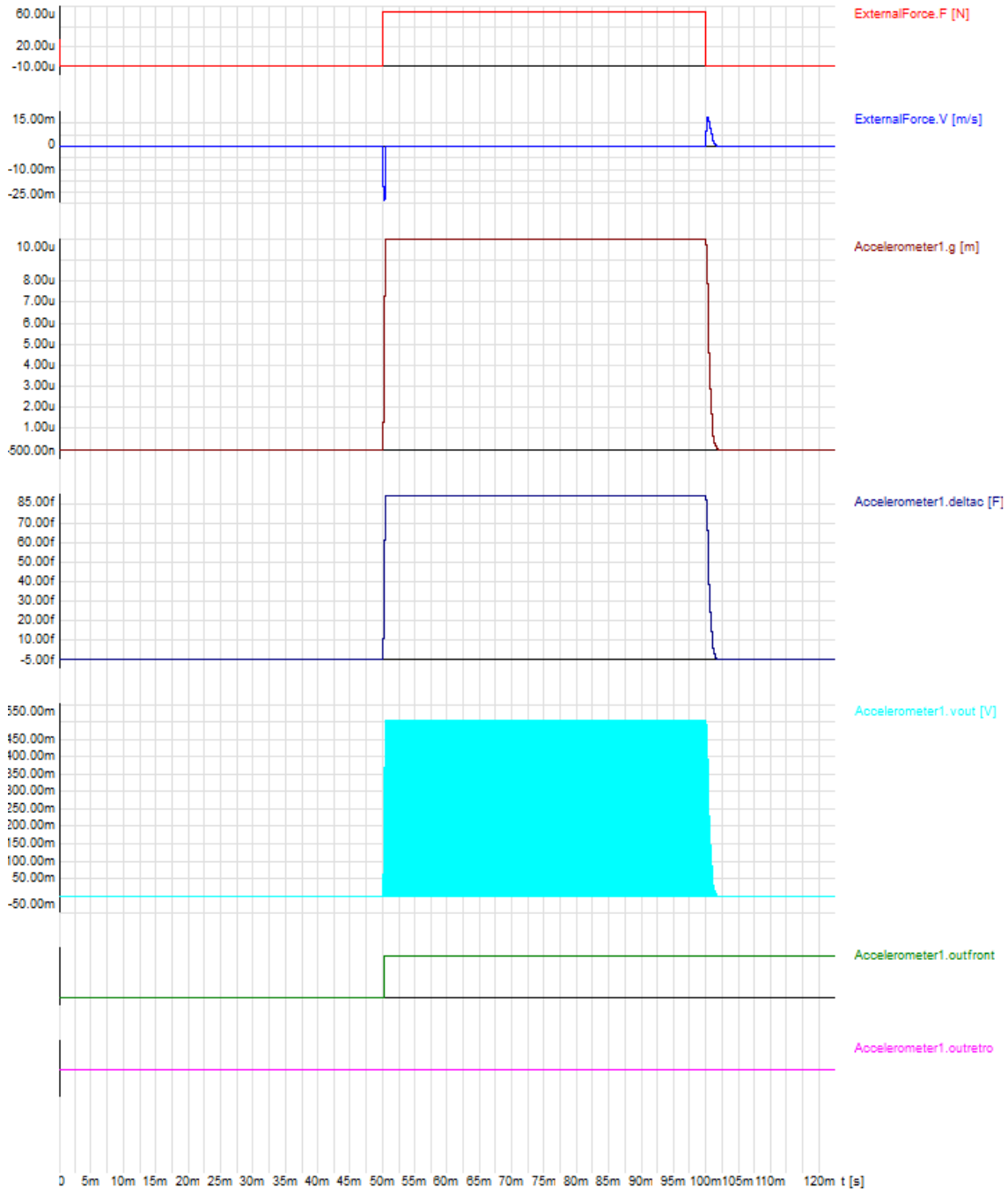


We can observe that the displacement g is about 7.5 µm and the corresponding capacitance variation ΔC is 65 fF. The maximum output voltage V_O is equal to about 380 mV and **FrontComparator** output is '1'. The

electrostatic force is a step with a maximum value of about $16 \cdot 10^{-9}N$. The Rise-Time is about 2 ms and all simulated values respect the expected results.

3.6.5 Simulation VHDL-AMS: Maximum Positive Acceleration

Since the maximum positive acceleration is 10.96 g we have set an input step of 20 g. The figure below shows the main signal obtained:

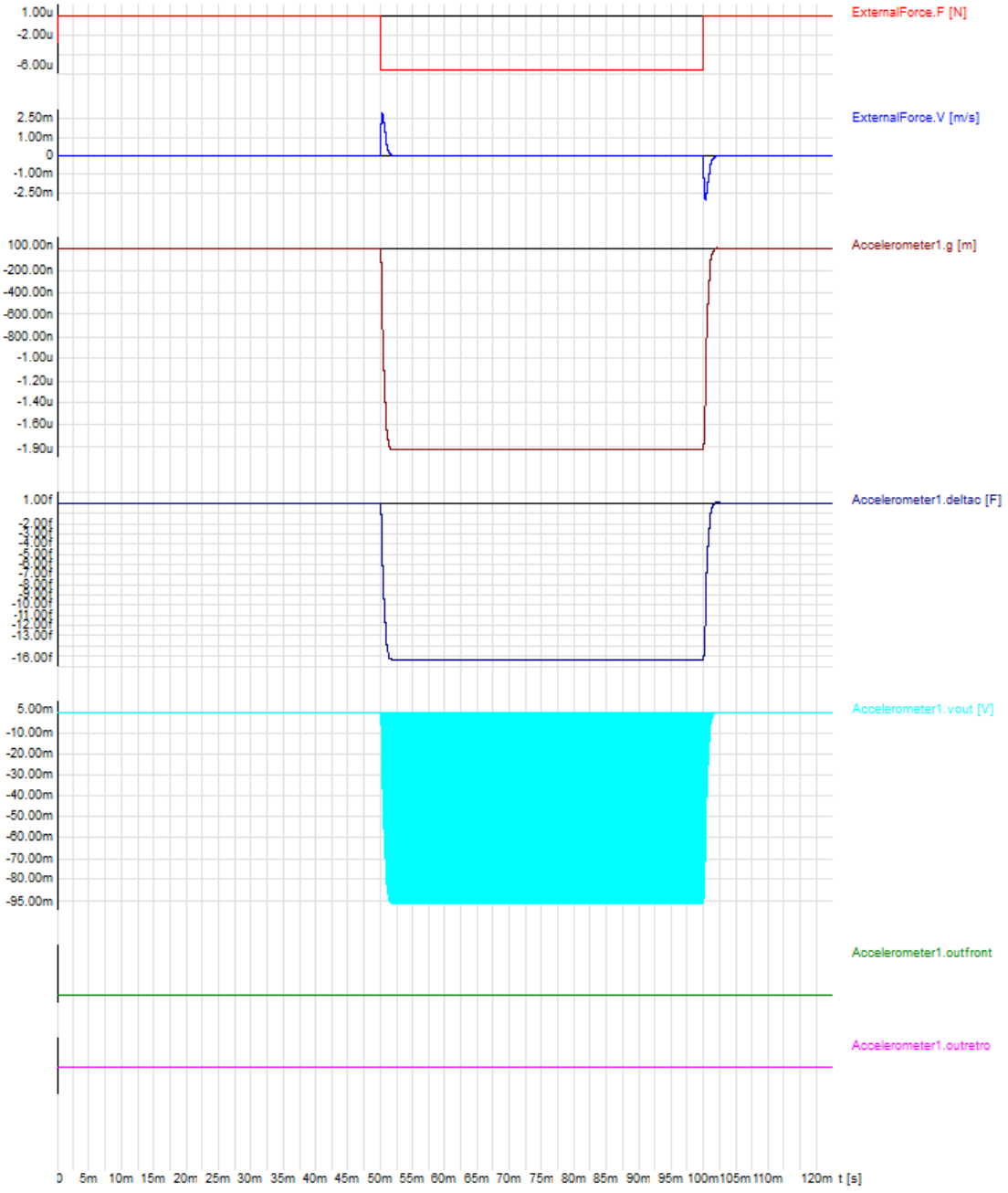


We can observe that the displacement g saturates to 10 µm and the corresponding ΔC is 85 fF, equal to the maximum value. The velocity has the behavior expected. The output voltage V_O is equal to about 500 mV and

FrontComparator output is '1'. The electrostatic force is a step with a maximum value of about $22 \cdot 10^{-9} N$. The Rise-Time is about 2 ms and all simulated values have the expected results.

3.6.6 Simulation VHDL-AMS: Negative Acceleration

Like the Simulink simulation, we have used a negative step acceleration of -2 g. The figure below shows the main signals obtained from the simulation.

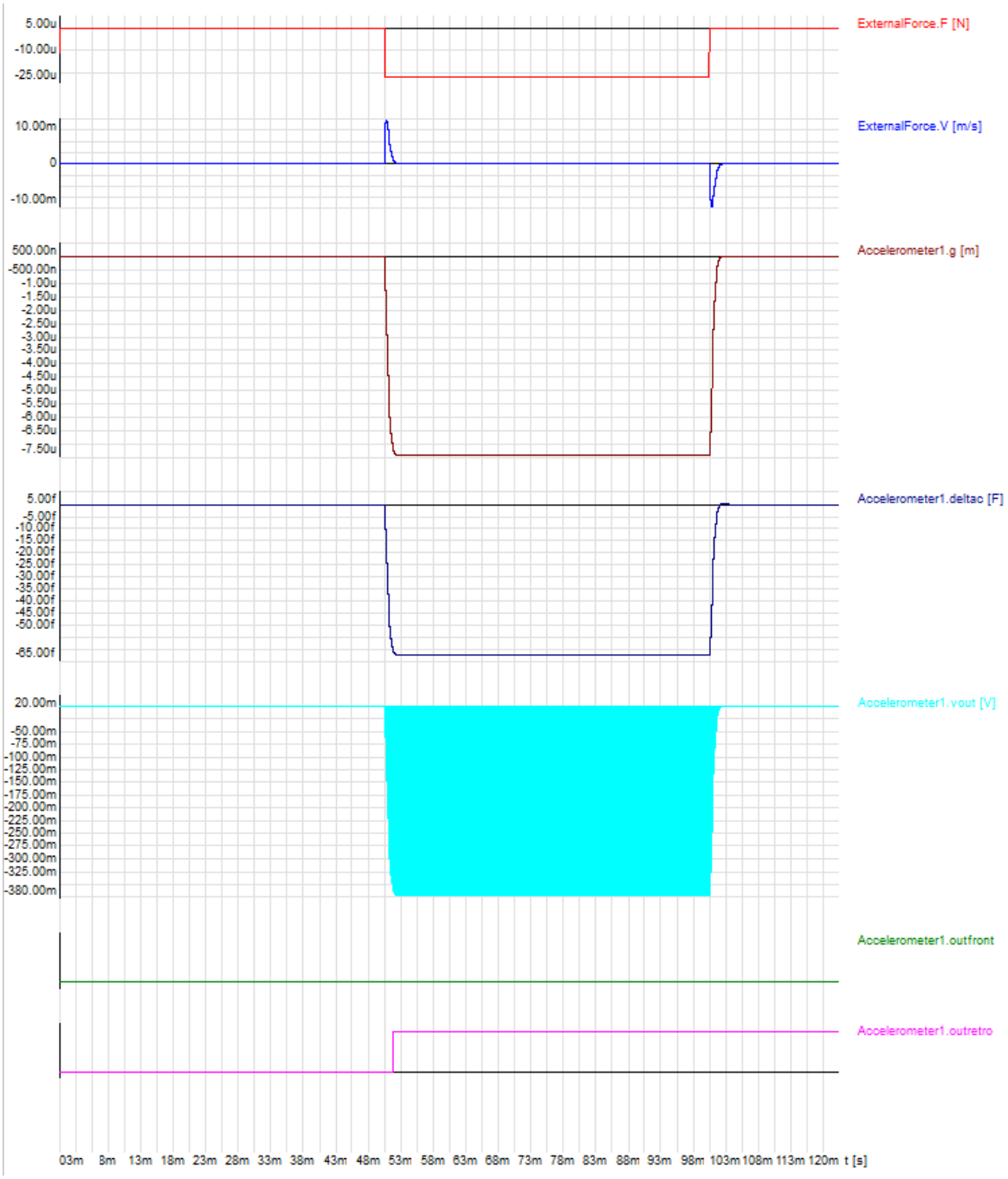


We can observe that the displacement g is about -1.9 µm and the corresponding absolute capacitance variation ΔC is 15.5 fF. The absolute maximum output voltage V_O is equal to about 95 mV. The comparators output

is '0'. The electrostatic force is a step with an absolute maximum value of about $4 \cdot 10^{-9} N$. The Rise-Time is about 2 ms and all simulated values respect the expected results.

3.6.7 Simulation VHDL-AMS: Negative Airbag Threshold Acceleration

In this case, we have tested the negative airbag threshold using step -8.1 g. The figure below shows the main signals obtained:

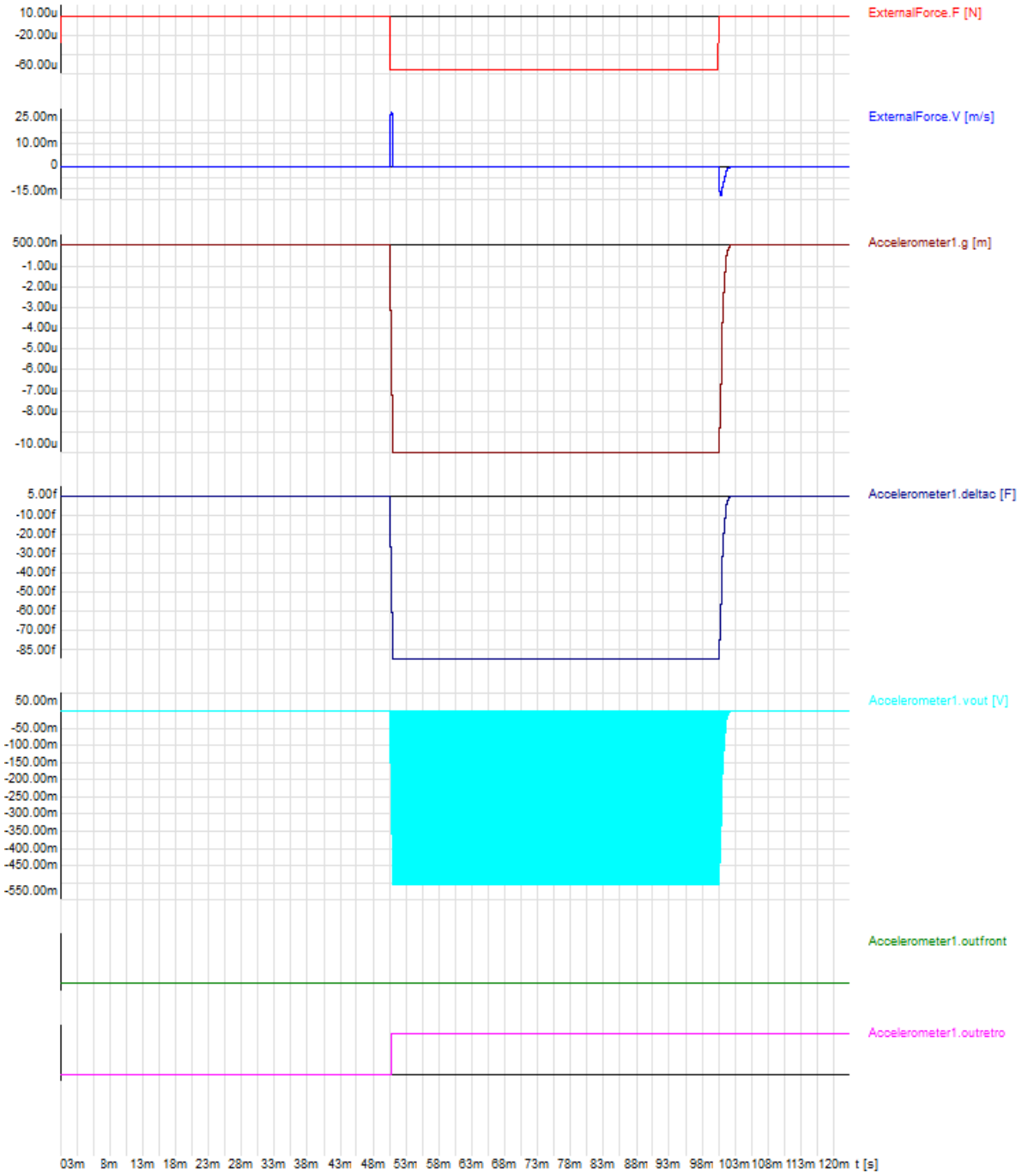


We can observe that the displacement g is about -7.5 µm and the corresponding capacitance variation ΔC is -65 fF. The absolute maximum

output voltage V_O is equal to about 380 mV and **RetroComparator** output is '1'. The electrostatic force is about $-16 \cdot 10^{-9} N$. The Rise-Time is 2 ms and all simulated values respect the expected results.

3.6.8 Simulation VHDL-AMS: Maximum Negative Acceleration

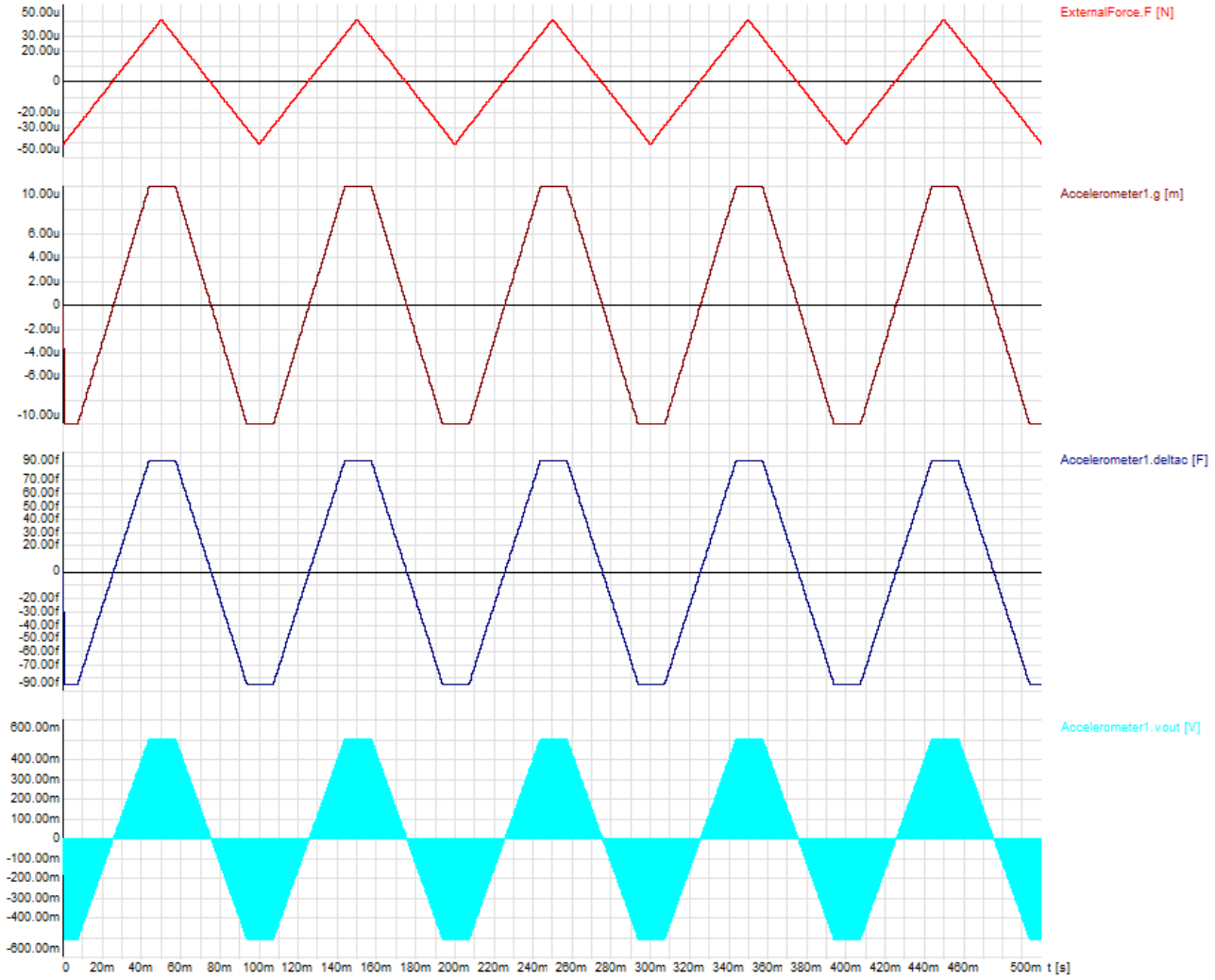
Since the maximum negative acceleration is -10.96 g we have set an input acceleration of -20 g. The figure below shows the main signals obtained:



We can observe that the displacement g saturates to -10 µm and the corresponding capacitance variation ΔC is -85 fF, equal to the negative maximum value. The velocity has the behavior expected. The absolute maximum output voltage V_O is equal to about 500 mV and **RetroComparator** output is '1'. The electrostatic force is a step with an absolute maximum value of about $22 \cdot 10^{-9} N$. The Rise-Time is about 2 ms and all simulated values respect the expected results.

3.6.9 Simulation VHDL-AMS: Triangular Input Acceleration

In order to observe the dynamic response of our system we have consider an input ramp acceleration from -15 g to 15 g with a frequency equal to 10 Hz.



As we can see, the displacement varies between -10 µm and 10 µm. It has the same shape of the input acceleration. The response of our system respects the expected results and it is comparable to Simulink.

3.6.10 Simulation VHDL-AMS: Brownian Noise

For the simulation of the Brownian noise, we have used a sinusoidal input as temperature noise with an amplitude of 0.1 Kelvin and frequency of 10 kHz. This choice is dependent from the tools available in Simplorer. The average value of the temperature is internally 298 K, the same as Simulink model.

3.7 Comparison of Results and Final Considerations

The results obtained from Simulink and VHDL-AMS are comparable. The developing of project on Simulink is more graphical and for this reason much clear for the debugging. Here the system is created by a lot of simple blocks connected together. With the VHDL-AMS the description is more abstract and more difficult to debug but thanks to the language description approach we can develop detailed and customized model quickly. MatLab on the other hand has a very complete environment where we have also the possibility to integrate the VHDL-AMS code, develop custom Simulink block etc. Our final consideration is that there is not the best solution because every approach has its strengths and its flaws, all depends on your needs. If you have a lot of equations inside your system probably the better choice could be the VHDL-AMS because every equation need only one row of code. On the other hand if you need a more graphical description with a lot of types of functions and operators Simulink can offer you a very complete environment.

4.0 Ending

The accelerometer project give us excellent results and possible future improvements could be:

- Adding the Y and Z axis for 3D acceleration measurements. This improvement could be expand the applications field;

- A more detailed description considering also the physical realization, fabrication processes and the packaging. This could bring other problem that have to be included inside the model.

Each improvement needs mathematical equations that must be inserted inside the model description.

Bibliography

- www.aci.it;
- www.quattroruote.it;
- en.wikipedia.org;
- VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multi-Discipline Systems. F. Pêcheux, C. Lallement, Member, IEEE, and A. Vachoux, Member, IEEE;
- Analog and Mixed-Signal Modeling Using the VHDL-AMS Language. 36th Design Automation Conference New Orleans, June 21-25, 1999;
- SENSING AND CONTROL OF MEMS ACCELEROMETERS USING KALMAN FILTER. KAI ZHANG Bachelor Degree of Science of Electronics in Radio-Communication, East China Normal University, China. June, 2000;
- Introduction To The VHDL-AMD Modeling Language. Scott Cooper,Mentor Graphics,Presented 13 November 2007,Westminster, Colorado;
- INCLINOMETRI IN TECNOLOGIA MEMS Relatore: prof. Leopoldo Rossetto, Laureando: Matteo De Biasi;
- Sviluppo di applicazioni basate su Wiimote per scopi didattici, Laureando:Daniele Rucatti, Relatore:prof.ssa Giada Giorgi;
- The Matlab/Simulink Modeling and Numerical Simulation of an Analogue, Capacitive Micro-Accelerometer. Part 1: open loop,Teodor Lucian Grigorie;
- Auto-calibration of capacitive MEMS accelerometers based on pull-in voltage,L. A. Rocha · R. A. Dias · E. Cretu ·L. Mol · R. F. Wolffenbuttel;