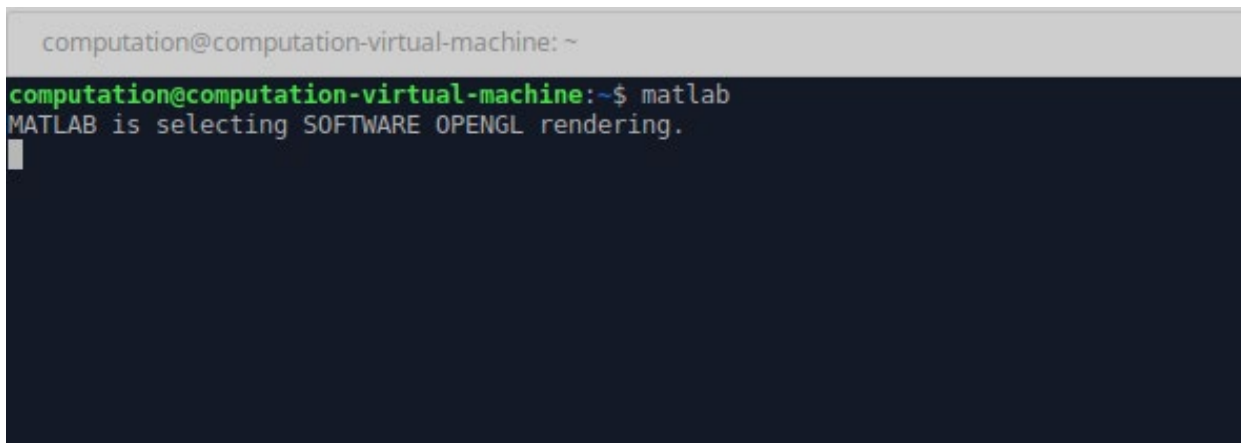


IMACS User Guide for Software-in-the-Loop (SiL) simulation and Finding WCET of Application Task

Section I: SiL implementation for sequential configuration

- open MATLAB: To open MATLAB in the IMACS virtual machine, run the following command in the new terminal.

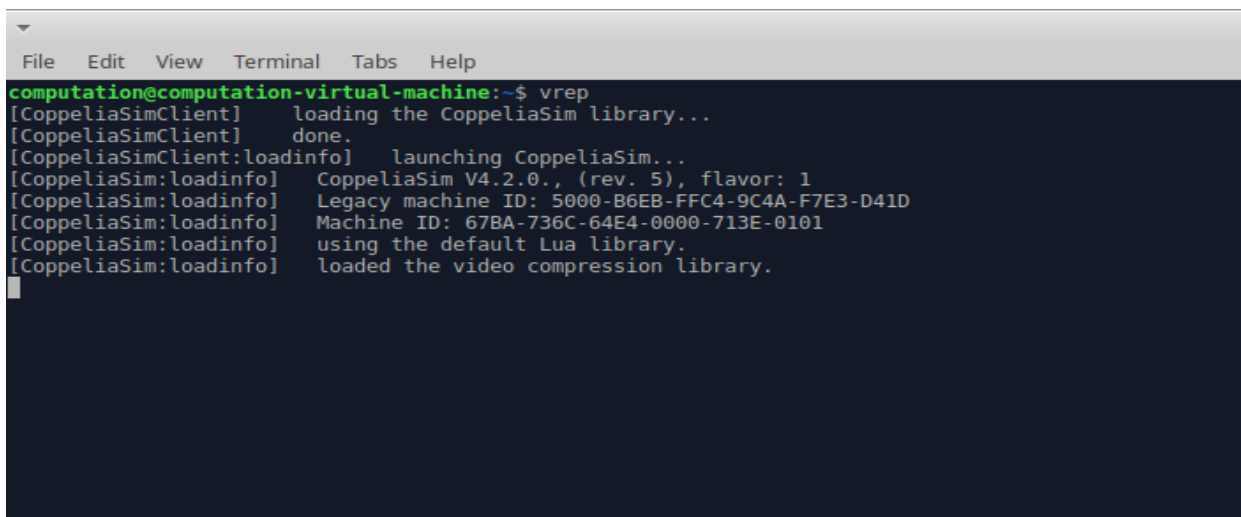
```
$ matlab
```

A terminal window with a dark background. The prompt is 'computation@computation-virtual-machine: ~'. The user enters 'matlab' and the output is 'MATLAB is selecting SOFTWARE_OPENGL rendering.'

```
computation@computation-virtual-machine: ~  
computation@computation-virtual-machine:~$ matlab  
MATLAB is selecting SOFTWARE_OPENGL rendering.
```

- open CoppeliaSim: to open the CoppeliaSim, run the following command in a new terminal.

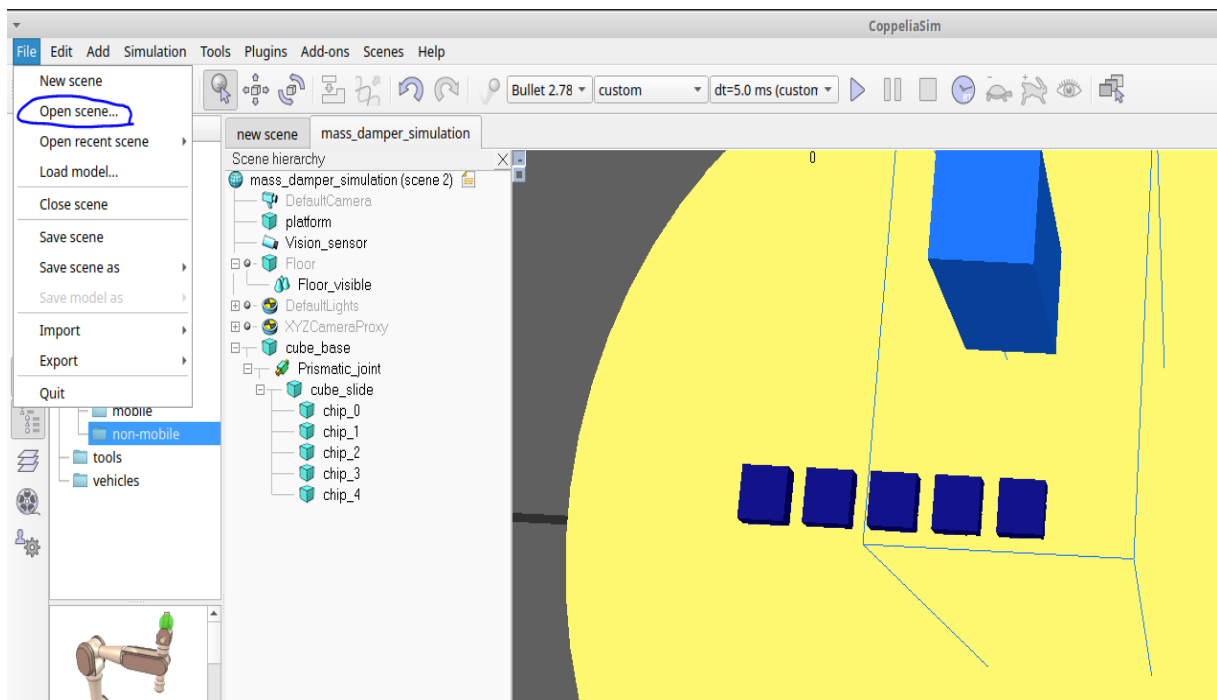
```
$ vrep
```

A terminal window with a dark background and a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is 'computation@computation-virtual-machine:~\$'. The user enters 'vrep' and the output shows CoppeliaSim loading libraries and launching.

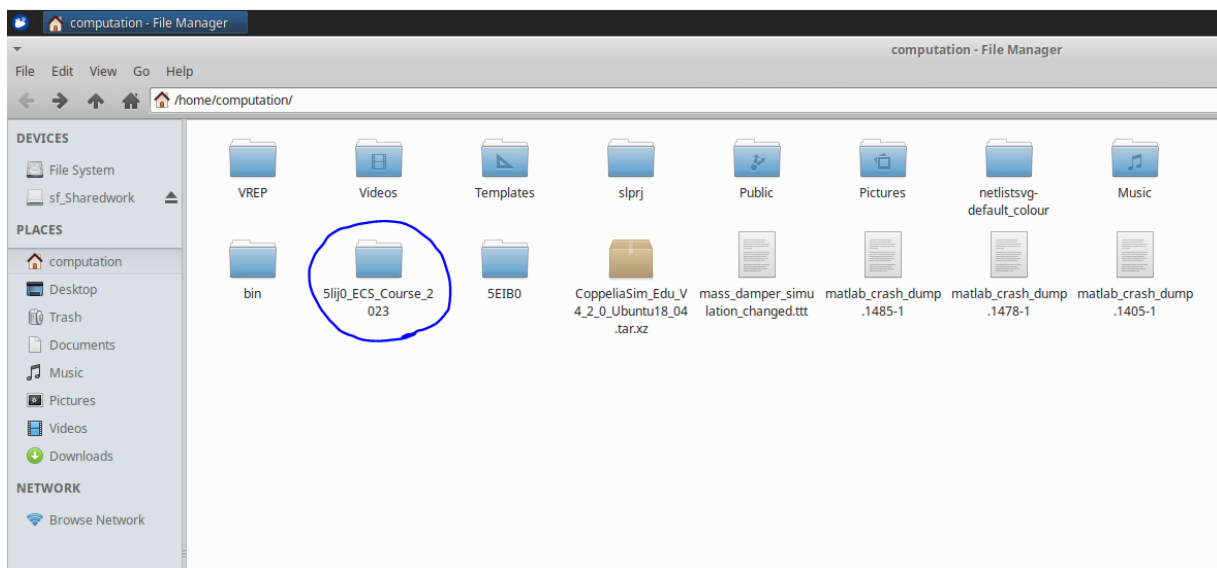
```
computation@computation-virtual-machine:~$ vrep  
[CoppeliaSimClient] loading the CoppeliaSim library...  
[CoppeliaSimClient] done.  
[CoppeliaSimClient:loadinfo] launching CoppeliaSim...  
[CoppeliaSim:loadinfo] CoppeliaSim V4.2.0., (rev. 5), flavor: 1  
[CoppeliaSim:loadinfo] Legacy machine ID: 5000-B6EB-FFC4-9C4A-F7E3-D41D  
[CoppeliaSim:loadinfo] Machine ID: 67BA-736C-64E4-0000-713E-0101  
[CoppeliaSim:loadinfo] using the default Lua library.  
[CoppeliaSim:loadinfo] loaded the video compression library.
```

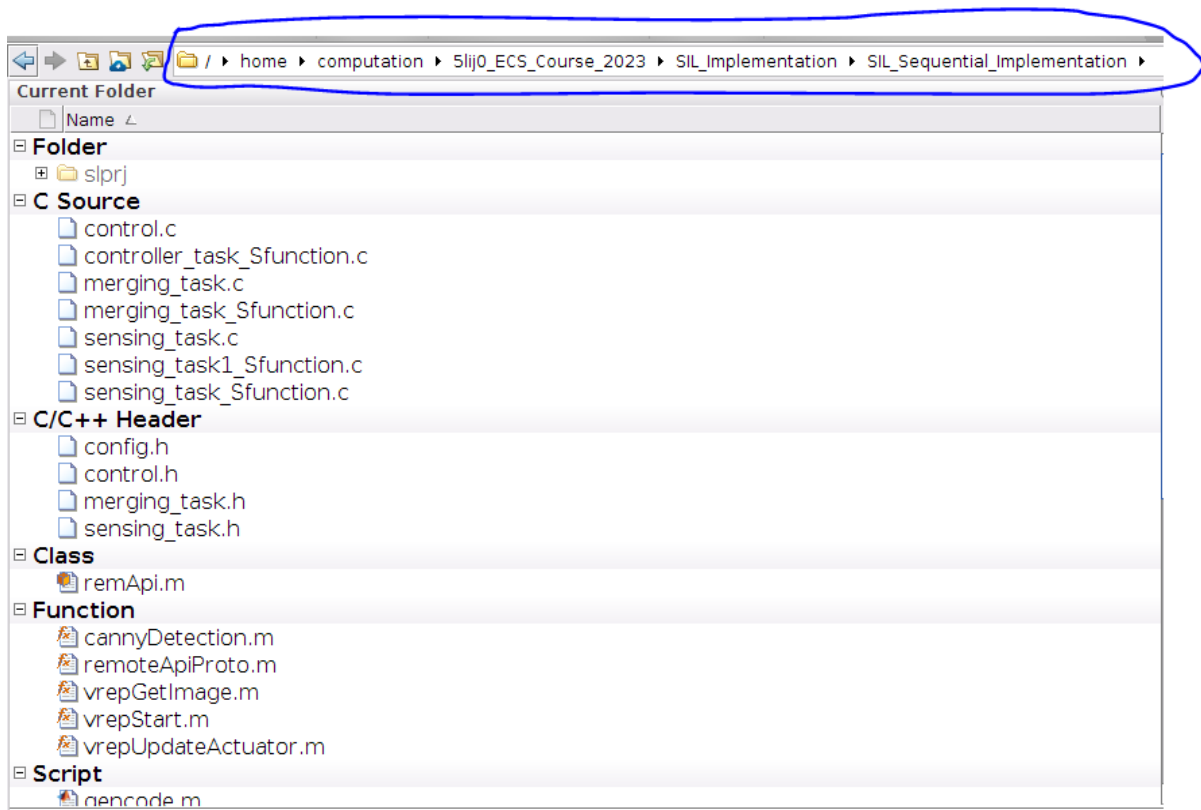
- open CoppeliaSim scene for the vision-in-the-loop system.
 - go to File→Open scene→ select the following directory
/home/computation/5lij0_ECS_Course_2023/SIL_Implementation/SIL_Sequential_I
mplementation/

open **mass_damper_simulation.ttt**, check the following screenshot for reference.

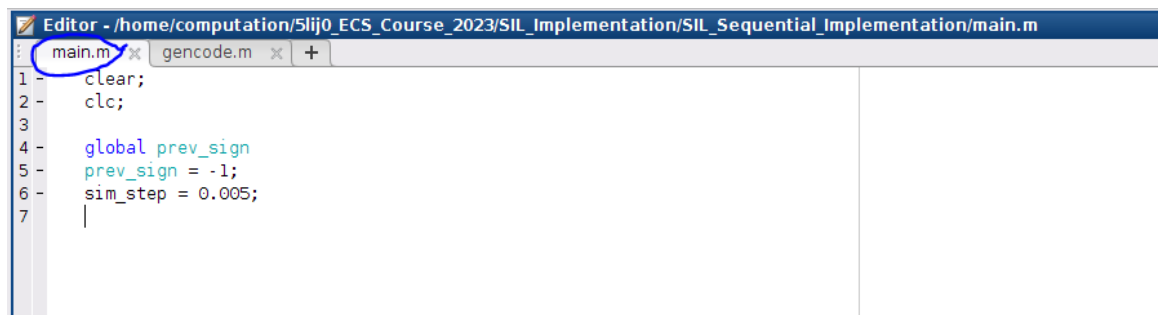


- open the folder for SiL sequential implementation.
 - go to the following folder path and open in MATLAB
computation→5lij0_ECS_Course_2023→SIL_Implementation→SIL_Sequential_Implementation





- open the `main.m` file and run the script. This file includes the initialization, which is required for the SiL simulation.



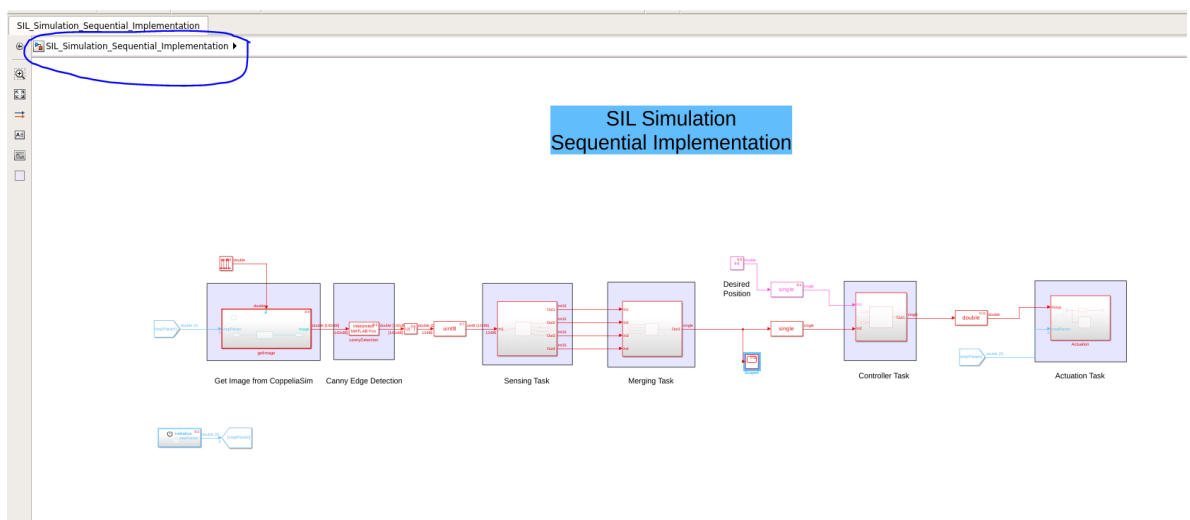
- open the `gencode.m` file and run the script. This file uses legacy code tool functionality, integrating the existing C or C++ functions into MATLAB Simulink. Once this `gencode.m` runs, it creates the S-function block inside the Simulink model.

```

Editor - /home/computation/5lij0_ECS_Course_2023/SiL_Implementation/SiL_Sequential_Implementation/gencode.m
main.m x gencode.m x +
1
2 %% Sensing Task
3
4
5 def_hough = legacy_code('initialize');
6 def_hough.SourceFiles = {'sensing_task.c'};
7 def_hough.HeaderFiles = {'sensing_task.h'};
8 def_hough.SFunctionName = 'sensing_task_Sfunction';
9 def_hough.OutputFcnSpec = 'void sensing_task(uint8 u1[12496], int16 y1[1], int16 y2[1],int16 y3[1], int16 y4[1])';
10 def_hough.IncPaths = {'../CompSOC_ec_target/files'};
11 legacy_code('sfcn_cmex_generate', def_hough);
12 legacy_code('sfcn_tlc_generate', def_hough);
13 legacy_code('compile', def_hough);
14 % legacy_code('slblock_generate', def_hough);|
15
16 %% Merging Task
17
18 %
19 def_merging = legacy_code('initialize');
20 def_merging.SourceFiles = {'merging_task.c'};
21 def_merging.HeaderFiles = {'merging_task.h'};
22 def_merging.SFunctionName = 'merging_task_Sfunction';
23 def_merging.OutputFcnSpec = 'single y1 = merging_task(int16 u1[1], int16 u2[1], int16 u3[1], int16 u4[1])';
24 def_merging.IncPaths = {'../CompSOC_ec_target/files'};
25 legacy_code('sfcn_cmex_generate', def_merging);
26 legacy_code('sfcn_tlc_generate', def_merging);
27 legacy_code('compile', def_merging);
28 % legacy_code('slblock_generate', def_merging);
29 %% Controller Task
30
31
32 def_control = legacy_code('initialize');
33 def_control.SourceFiles = {'control.c'};
34 def_control.HeaderFiles = {'control.h'};
35 def_control.SFunctionName = 'controller_task_Sfunction';
36 def_control.OutputFcnSpec = 'single y1 = PID_controller(single u1, single u2)';
37 def_control.IncPaths = {'../CompSOC_ec_target/files'};
38 legacy_code('sfcn_cmex_generate', def_control);
39 legacy_code('sfcn_tlc_generate', def_control);

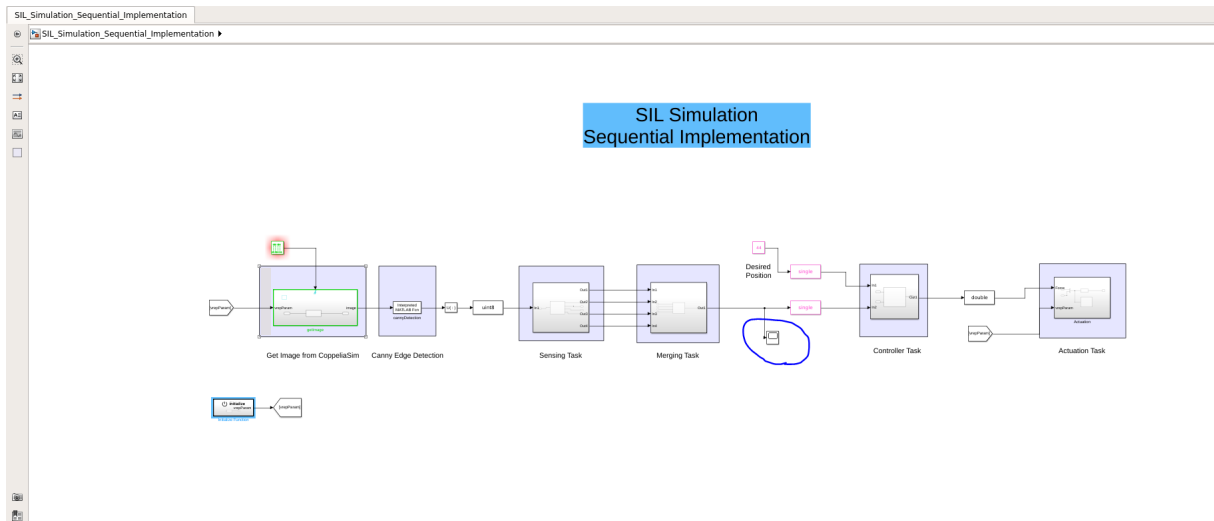
```

- open the SiL simulation file named as **SiL_Simulation_Sequential_Implementation.slx**
Students can find this file in the following directory:
computation→5lij0_ECS_Course_2023→SiL_Implementation→SiL_Sequential_Implementation
- run the **SiL_Simulation_Sequential_Implementation.slx** simulink model. Make sure the CoppeliaSim scene is opened in the software. The Simulink model contains each application task used in the sequential implementation. Sensing, merging, and controller task. Students can find out the execution time of each task independently using PiL (Processor-in-the-Loop) simulation. Check the next steps for finding the execution time of each task.



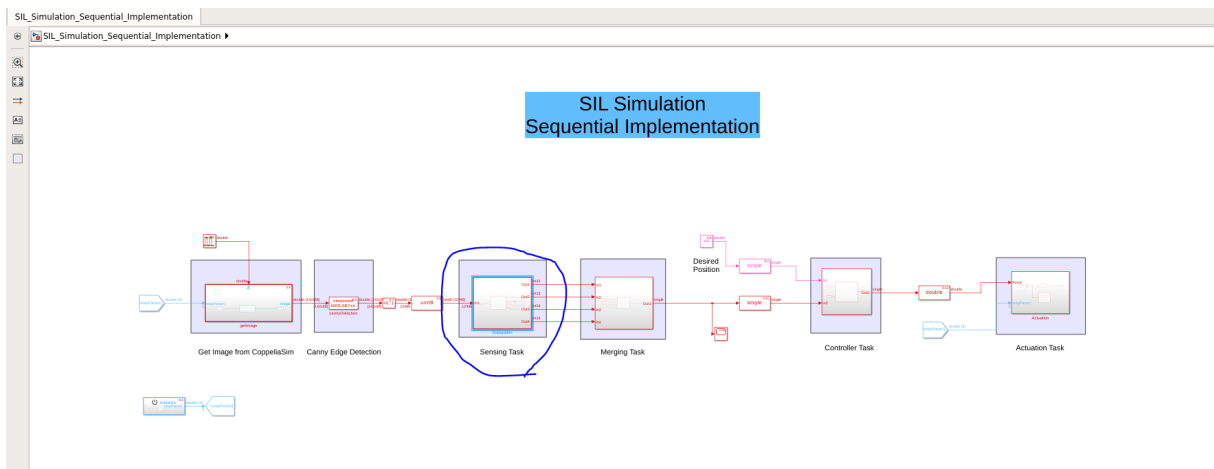
- visualize the output of the SiL simulation in CoppeliaSim. Also, check the current position

returned from the merging task in the scope linked to that block.

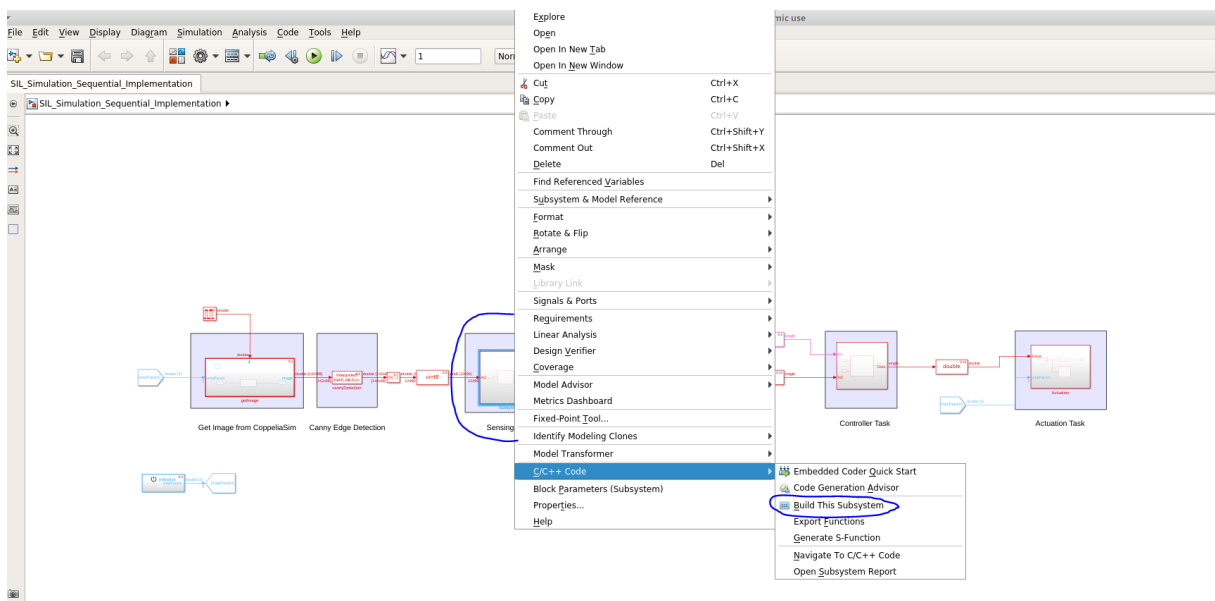
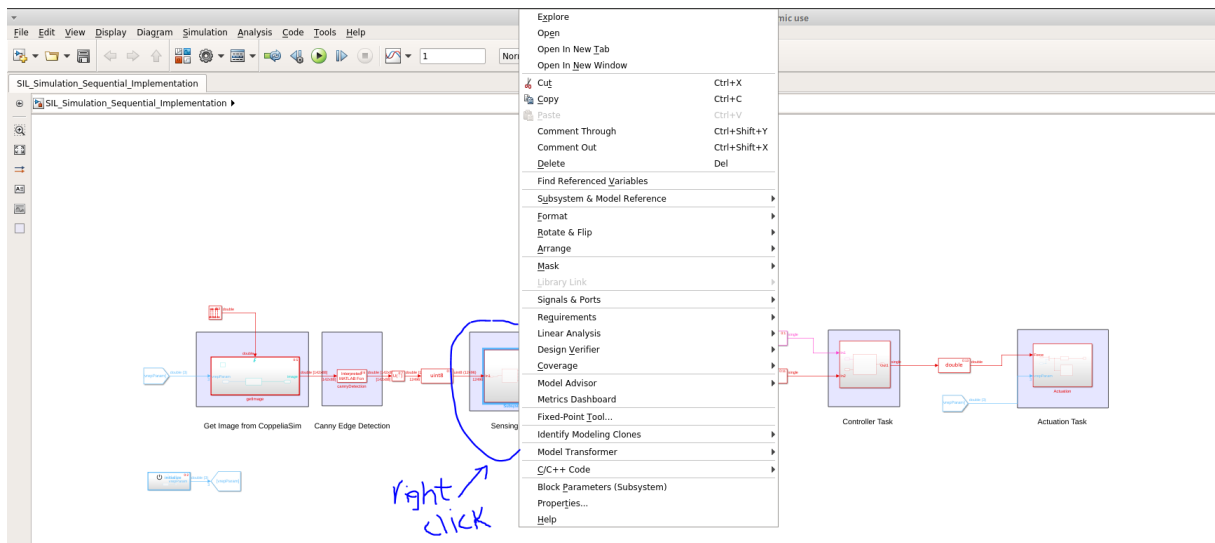


Subsection I.I: Measuring the execution time of each application task in the sequential configuration

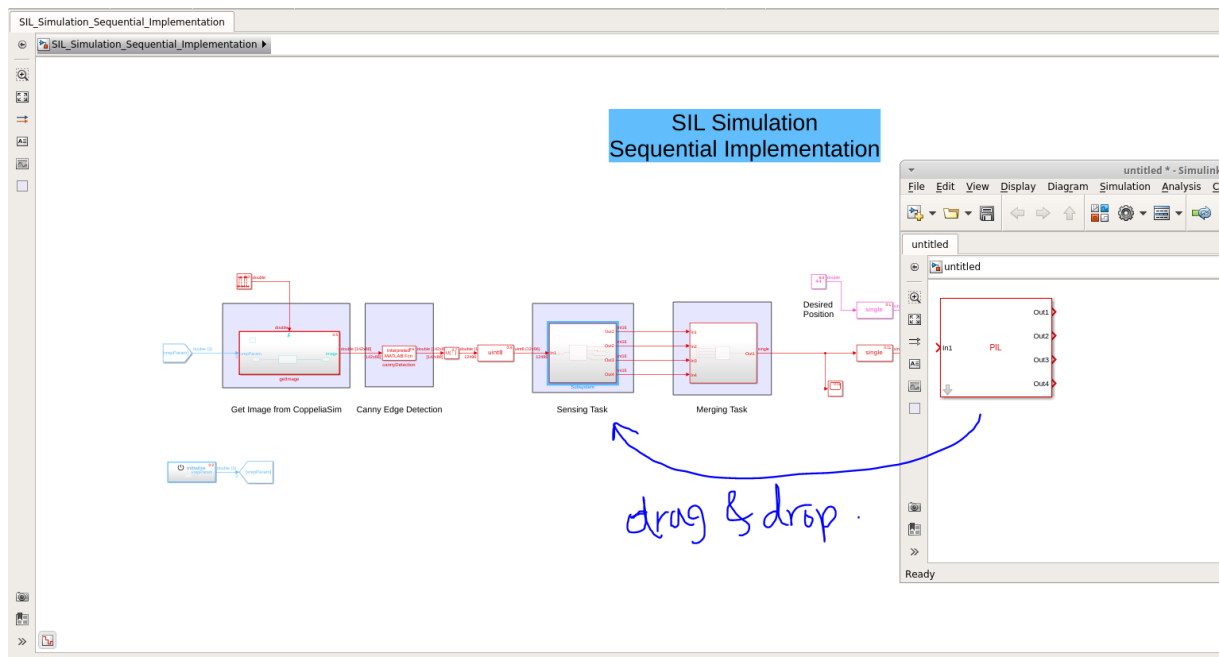
- after running the SiL simulation for sequential configuration, students need to find out the WCET (Worst-Case-Execution-Time) time of each task. WCET can find out using the following steps:
 - in the **SIL_Simulation_Sequential_Implementation.slx** model, check the first task, which is the sensing task, highlighted in the following screenshot.



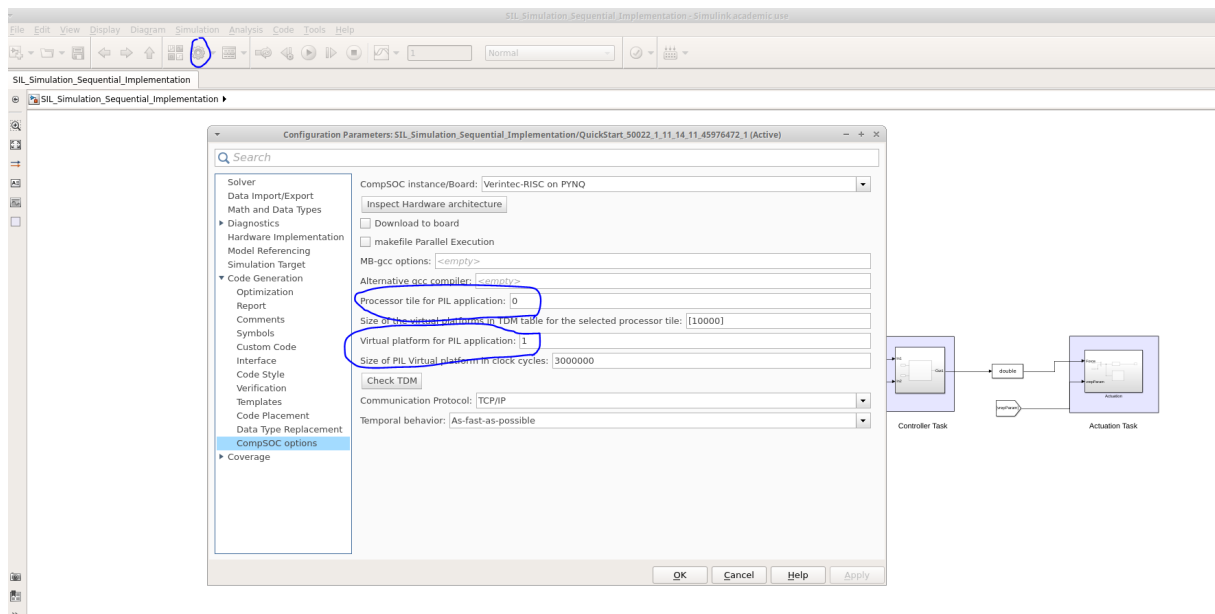
- right click on the block and select the C/C++ code option and click on build this subsystem. Check the following screenshot for reference.



- once, Build process completed successfully, it will generate the PIL block, check the following screenshot:



- after generating the PIL block, replace the existing sensing task block with the generated PIL block. Drag and drop will work to use the PIL block in place of the sensing task block. Make sure to comment existing sensing block and keep it aside. Don't delete the block; otherwise, regeneration of the block will be required.
- before running the simulation, check the configuration setting highlighted in the screenshot.



- the configuration settings for checking the WCET of each task are as follows:
 - Processor tile for PIL application : 0
 - Virtual platform for PIL application: 1
- sensing task requires approximately 54KB of memory, and other task requires less than 32KB of memory, therefore before running the simulation, use the

vep_config.txt file from the following directory:

computation→5lij0_ECS_Course_2023→SIL_Implementation→SIL_Sequential_Implementation

- copy the contents from the above-mentioned **vep_config** file into the main **vep-config.txt** file, which is inside the following folder path :

/home/computation/CompSOC_ec_target/CompSOC_ec/+CompSOC_ec/vep-config.txt

- The contents in the vep_config file should be as follows:

```
File Edit Search View Document Help
##
## you can comment out & modify the "on tile ... " lines
##
## - partitions can only have a memory size of 32K or 64K
## - the size of all partitions on a tile must be at most 96K
##   (i.e. 128K including the system application)
## - default memory is 32K, default stack is 4K
##
## use /opt/riscv/bin/riscv32-unknown-elf-size [-A] *.elf
## to analyse a partition's memory usage
##
## memory and stack allocation
##
on tile 0 partition 1 has 64K memory and 4K stack
on tile 0 partition 2 has 32K memory and 4K stack
#on tile 0 partition 3 has 32K memory and 4K stack
on tile 1 partition 1 has 64K memory and 4K stack
on tile 1 partition 2 has 32K memory and 4K stack
#on tile 1 partition 3 has 32K memory and 4K stack
#on tile 2 partition 1 has 64K memory and 4K stack
#on tile 2 partition 2 has 32K memory and 4K stack
#on tile 2 partition 3 has 32K memory and 4K stack
##
## scheduling
##
## - max 3 slots per tile
## - it is allowed to not schedule anything on a tile
## - a partition can have more than one slot
## - the system partition always get a slot of 5000 cycles
##   at the end: you don't have to add this
##
#on tile 0 next slot is for partition 1 with 200000 cycles
on tile 0 next slot is for partition 1 with 1200000 cycles
on tile 0 next slot is for partition 2 with 10000 cycles
#on tile 0 next slot is for partition 3 with 377000 cycles
on tile 1 next slot is for partition 1 with 1200000 cycles
on tile 1 next slot is for partition 2 with 389000 cycles
#on tile 1 next slot is for partition 3 with 372000 cycles
#on tile 2 next slot is for partition 1 with 700000 cycles
#on tile 2 next slot is for partition 2 with 289000 cycles
# on tile 2 next slot is for partition 2 with 10000 cycles
# on tile 2 next slot is for partition 3 with 100000 cycles
```

- once all the above steps are completed then, open the two new terminal windows, connect the PYNQ board, and use the following command:

- Terminal 1 :
 - cd tutorial
 - cd monitoring-tools/
 - sudo ./channel_cheap_bridge 0 1 9878
- Terminal 2 :
 - cd tutorial
 - ./readout.sh

The image shows two terminal windows side-by-side. The left window shows the process of connecting to a remote host via SSH and running a command to create a network bridge. The right window shows the execution of a script that initializes memory and starts reading data.

```

computation@computation-virtual-machine: ~
computation@computation-virtual-machine:~$ ssh student@pato-board.local
ssh: connect to host pato-board.local port 22: Connection refused
computation@computation-virtual-machine:~$ ssh student@pato-board.local
student@pato-board.local's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.14.0-xilinx-00014-g14b3cc2178b6 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Feb 20 21:17:04 2023 from 10.42.0.1
student@pato-board:~$ cd tutorial
student@pato-board:~/tutorial$ cd monitoring-tools/
student@pato-board:~/tutorial/monitoring-tools$ sudo ./channel_cheap_bridge 0 1 9878
+Socket created
+bind on port: 9878 done
+ Listen done
-Waiting for incoming connections.

```

```

Terminal - computation@computation-virtual-machine:~
File Edit View Terminal Tabs Help

Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.14.0-xilinx-00014-g14b3cc2178b6 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

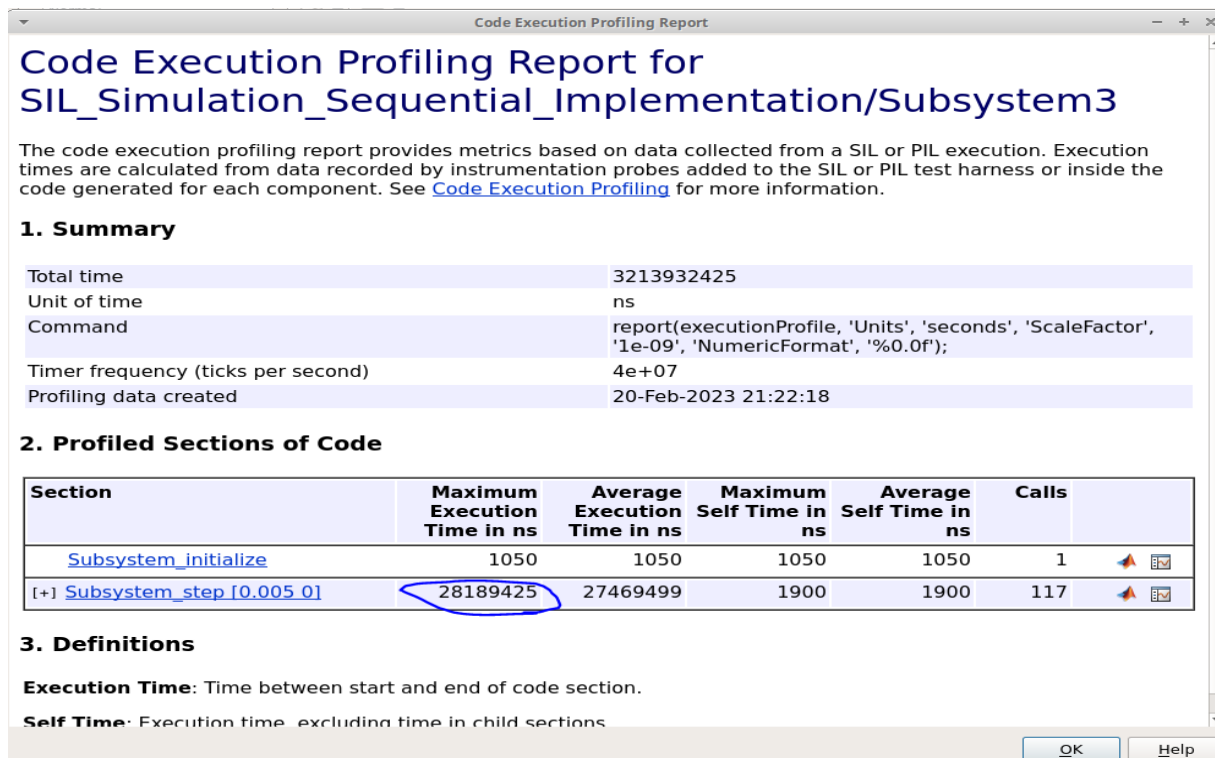
0 packages can be updated.
0 updates are security updates.

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Feb 21 04:59:55 2023 from 10.42.0.1
student@pato-board:~$ cd tutorial
student@pato-board:~/tutorial$ ./readout.sh
mem: 16 KB @ 0x0x80000000
Cheap stdout initialised
mem: 16 KB @ 0x0x80004000
Cheap stdout initialised
mem: 16 KB @ 0x0x80008000
Cheap stdout initialised
start reading

```

- run the **main.m** script again, and then run the **SIL_Simulation_Sequential_Implementation.slx** model. For every new execution of the Simulink model, make sure to run the main.m script before running the Simulink model; otherwise, it will give an error related to array dimensions.
- once the simulation is finished, it will generate the profiling report. In the report, the maximum execution time is the WCET of the particular task. Check the highlighted part from the following screenshot.



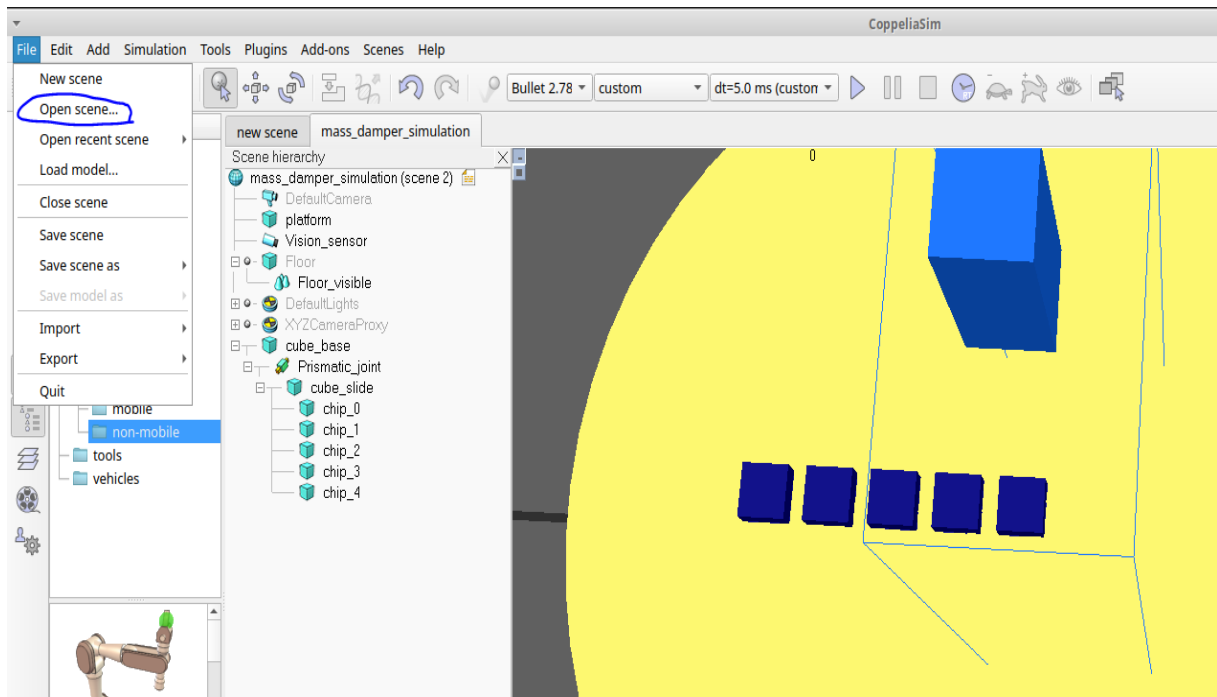
- repeat the above steps to find the WCET of each application task. To understand the entire process, explanations are given considering the sensing task. Students should follow the same steps for finding the WCET of other tasks, such as merging and controller tasks. No need to find the timing for the actuation task.

Note: Find the WCET of each task individually; it is not possible to find them all together. So follow the below steps to avoid errors:

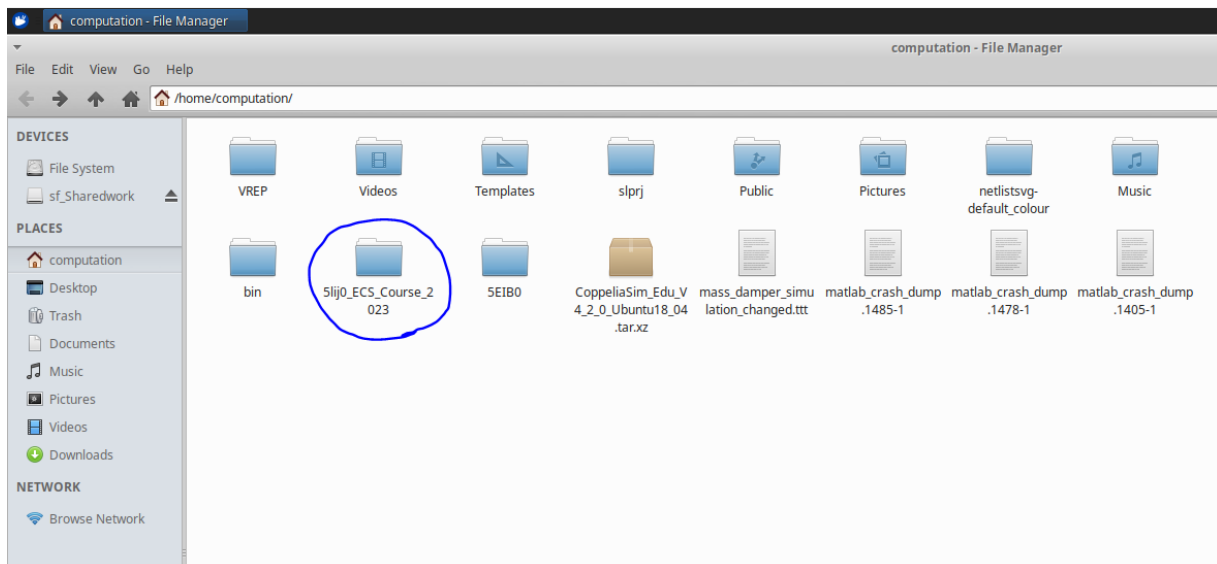
- 1) Generate the PIL block of a particular task for which WCET needs to find. Check the aforementioned steps.
- 2) Keep the original block, i.e., (S-function block aside as commented.) If it is not commented, then both the PIL and original blocks will run together and give the run-time error.
- 3) Other blocks should be as it is in the entire Simulink model.
- 4) Note down the timing as explained above.
- 5) Repeat the same step for other tasks.

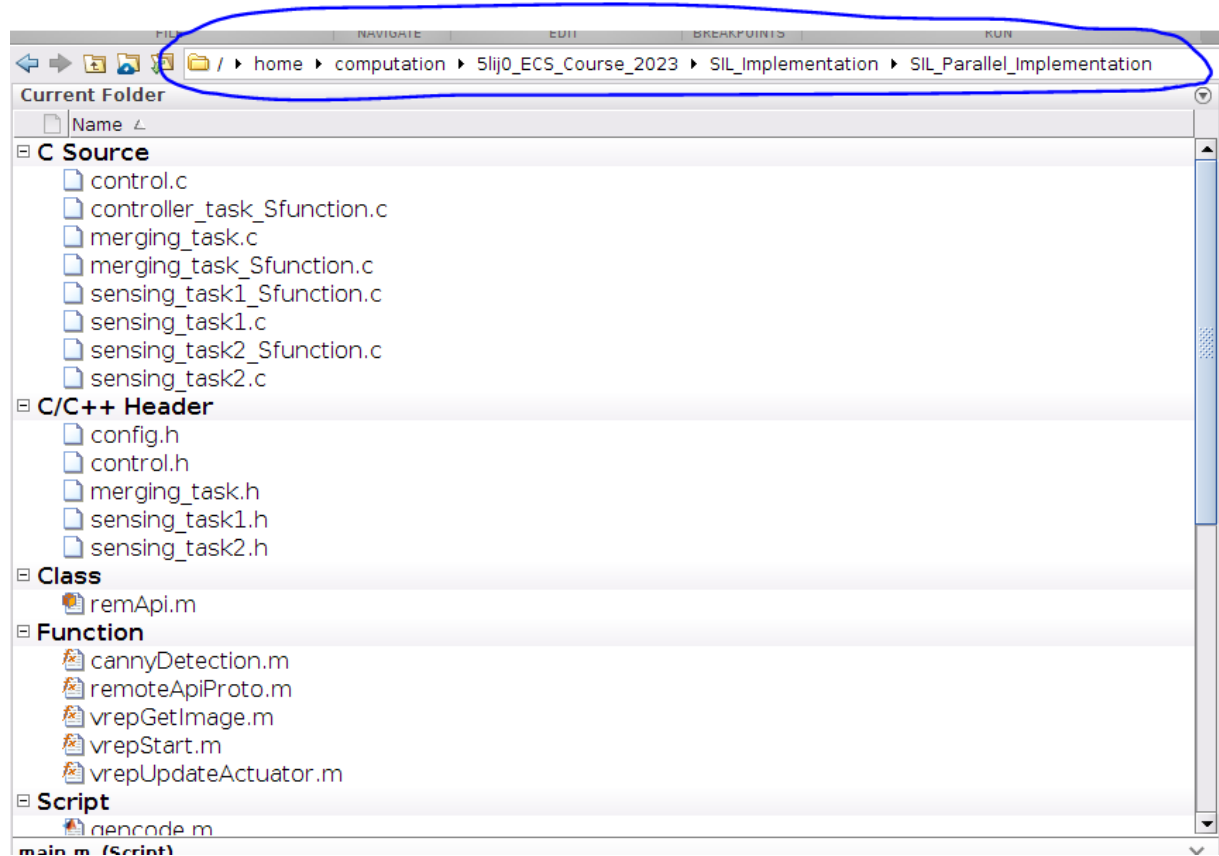
Section II: SiL implementation for parallel configuration

- open MATLAB and CoppeliaSim: Check Section I for opening MATLAB and CoppeliaSim. If the sequential part has been completed already, then no need to open MATLAB and CoppeliaSim software again.
- open CoppeliaSim scene for the mass-damper system.
 - go to File→Open scene→ select the following directory
/home/computation/5lij0_ECS_Course_2023/SIL_Implementation/SIL_Parallel_Implementation/
open **mass_damper_simulation.ttt**, check the following screenshot for reference.

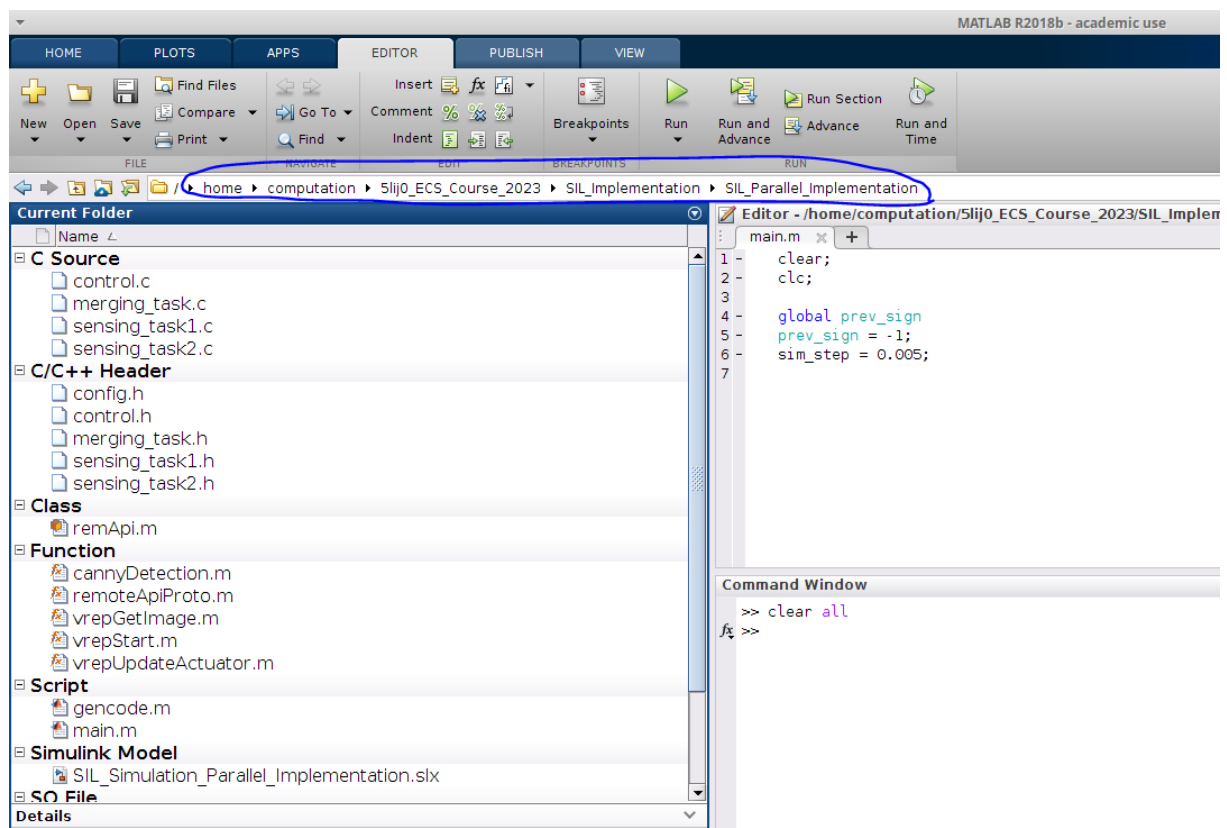


- open the folder for SiL parallel implementation.
 - go to the following folder path and open in MATLAB
 computation/5lij0_ECS_Course_2023/SiL_Implementation/SiL_Parallel_Implementation

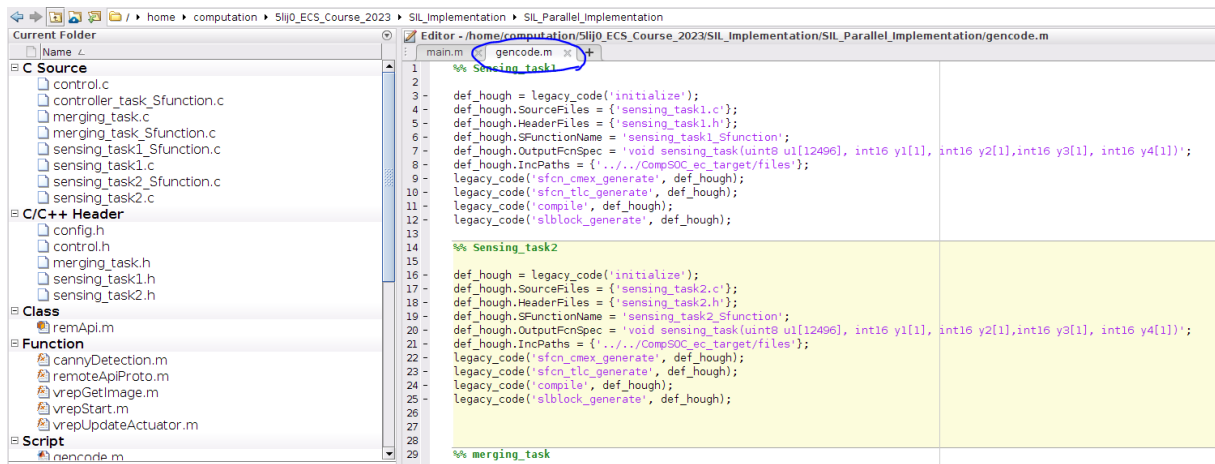




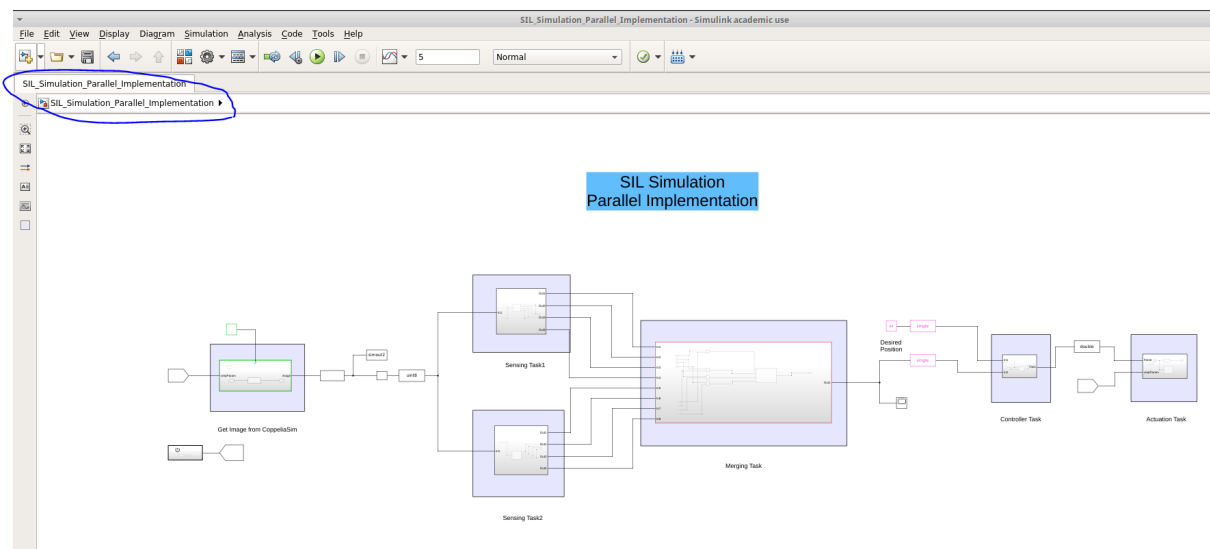
- open the main.m file and run the script. This file includes the initialization, which is required for the SiL simulation.



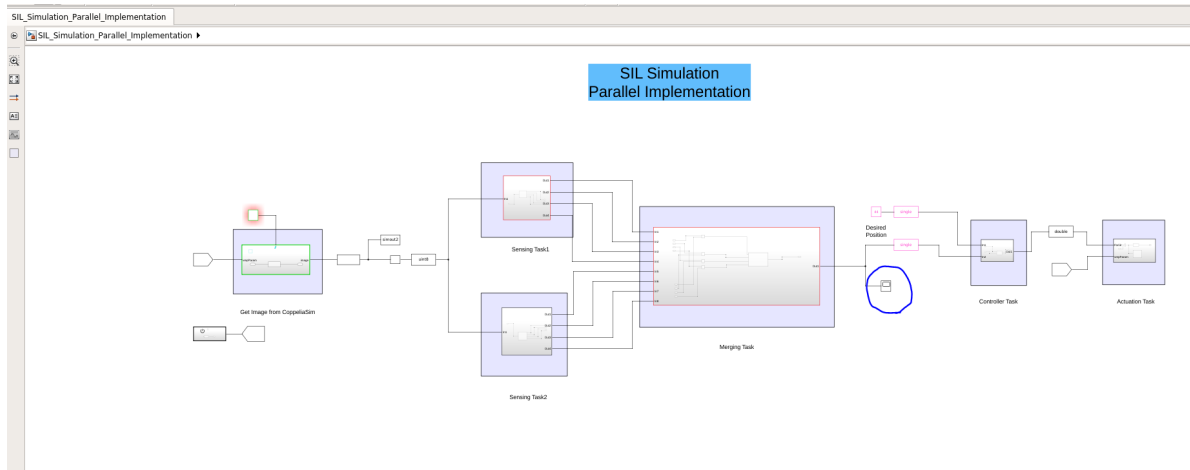
- open the gencode.m file and run the script. This file uses legacy code tool functionality, integrating the existing C or C++ functions into MATLAB Simulink. Once this gencode.m runs, it creates the S-function block inside the Simulink model.



- open the SiL simulation file named as **SIL_Simulation_Parallel_Implementation.slx**. Students can find this file in the following directory:
computation→5lij0_ECS_Course_2023→SIL_Implementation→SIL_Parallel_Implementation
- run the **SIL_Simulation_Parallel_Implementation.slx** simulink model. Make sure the CoppeliaSim scene is opened in the software. The Simulink model contains each application task used in the parallel implementation. Sensing task 1, sensing task 2, merging, and controller task. Students can find out the execution time of each task independently using PiL (Processor-in-the-Loop) simulation. Check the next steps for finding the execution time of each task.

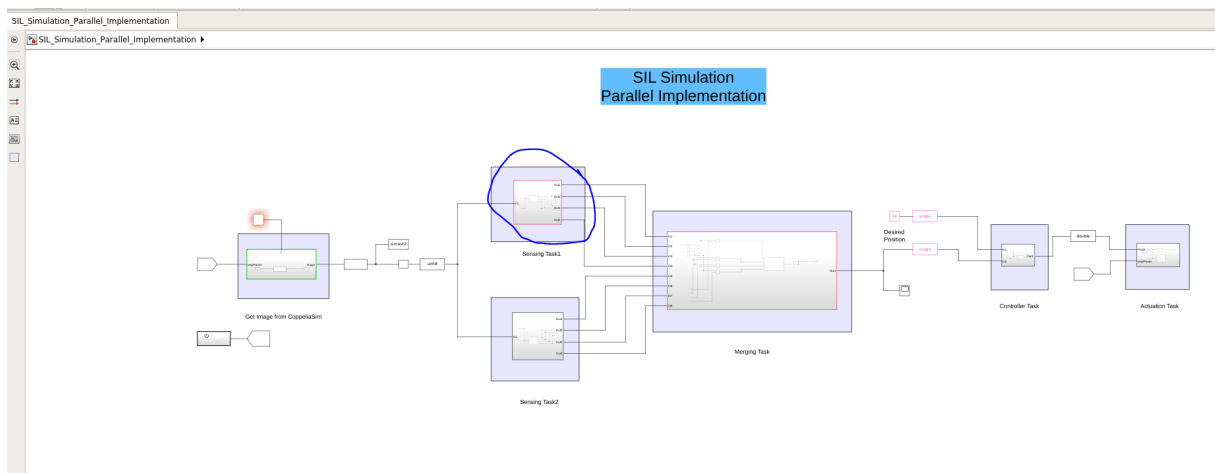


- visualize the output of the SiL simulation in CoppeliaSim. Also, check the current position returned from the merging task in the scope linked to that block.

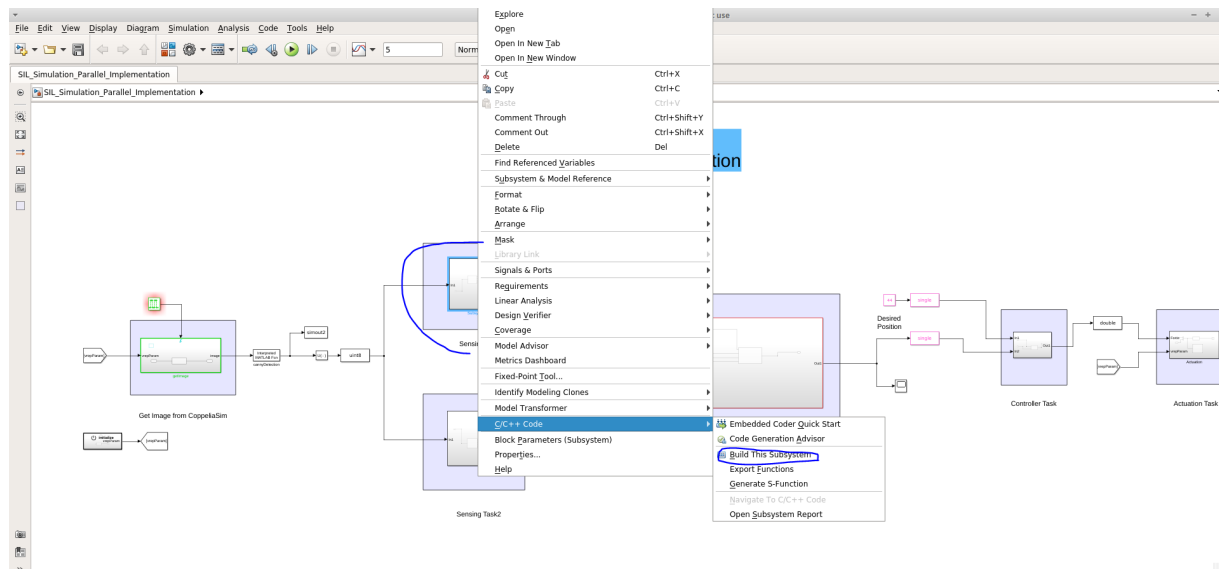


Subsection II.I: Measuring the execution time of each application task in parallel configuration

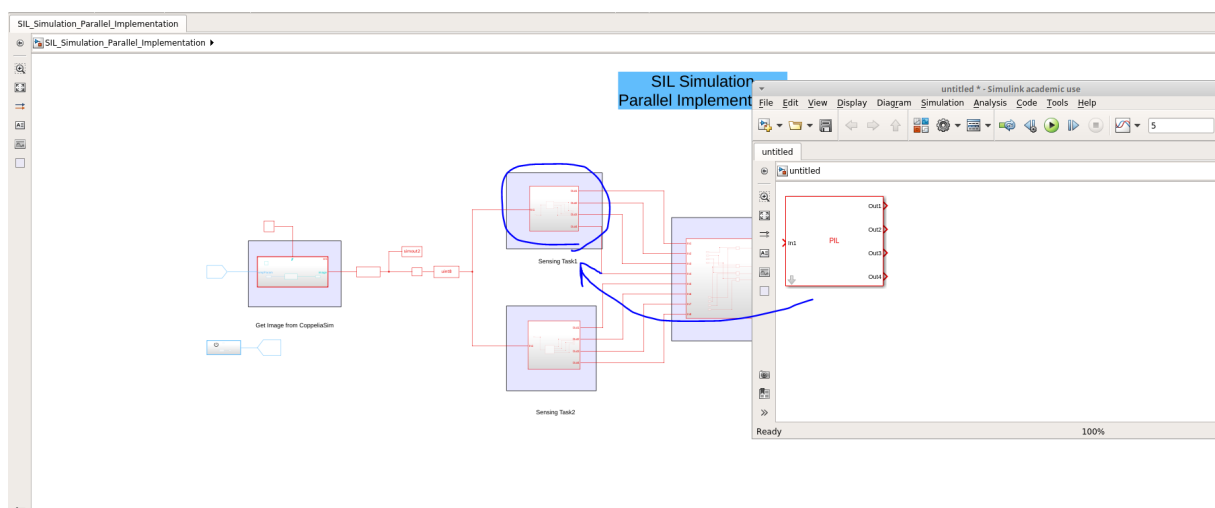
- after running the SiL simulation for parallel configuration, students need to find out each task's WCET (Worst-Case-Execution-Time) time. WCET can find out using the following steps:
 - in the **SIL_Simulation_Parallel_Implementation.slx** model, check the first task, which is the sensing task, highlighted in the following screenshot.



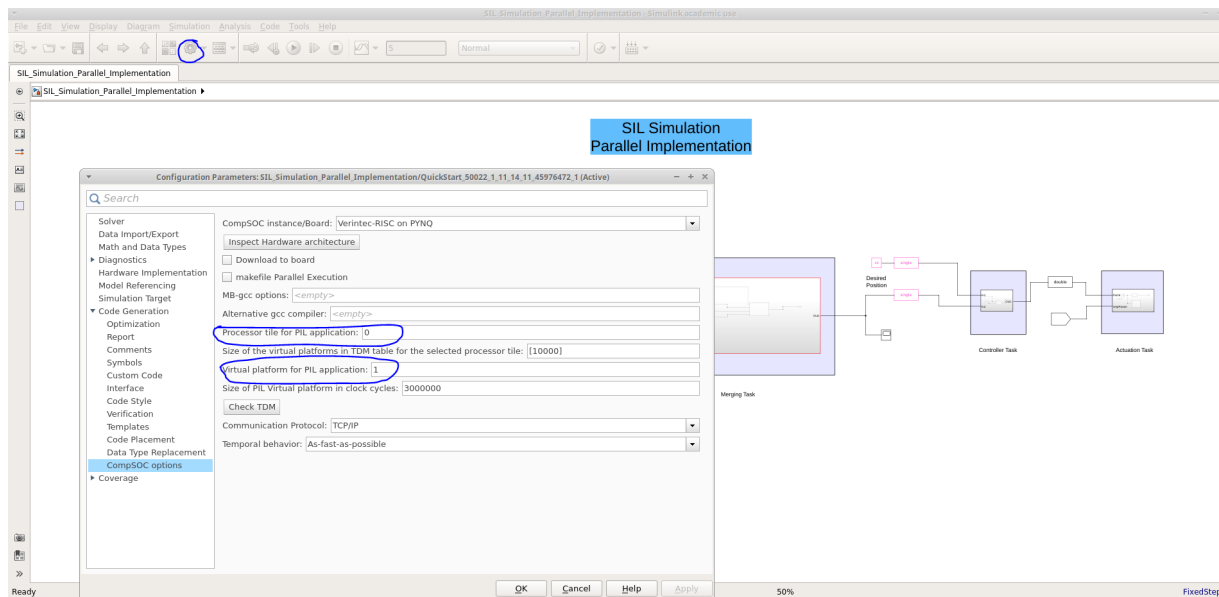
- Right-click on the block, select the C/C++ code option and click on build this subsystem. Check the following screenshot for reference.



- once, Build process is completed successfully, it will generate the PIL block; check the following screenshot:



- after generating the PIL block, replace the existing sensing task1 block with the generated PIL block. Drag and drop will work to use the PIL block in place of the sensing task1 block. Make sure to comment on existing sensing block 1 and keep it aside. Don't delete the block; otherwise, regeneration of the block will be required.
- before running the simulation, check the configuration setting highlighted in the screenshot.



- the configuration settings for checking the WCET of each task are as follows:
 - Processor tile for PIL application : 0
 - Virtual platform for PIL application: 1
- sensing task requires approximately 54KB of memory, and other task requires less than 32KB of memory, therefore before running the simulation, use the **vep_config.txt** file from the following directory:

computation→5lj0_ECS_Course_2023→SIL_Implementation→SIL_Parallel_Implementation

- copy the contents from above-mentioned **vep_config** file into the main **vep-config.txt** file, which is inside the following folder path :

/home/computation/CompSOC_ec_target/CompSOC_ec/+CompSOC_ec/vep-config.txt

- the contents in the vep_config file should be as follows:


```
File Edit Search View Document Help
##
## you can comment out & modify the "on tile ... " lines
##
## - partitions can only have a memory size of 32K or 64K
## - the size of all partitions on a tile must be at most 96K
##   (i.e. 128K including the system application)
## - default memory is 32K, default stack is 4K
##
## use /opt/riscv/bin/riscv32-unknown-elf-size [-A] *.elf
## to analyse a partition's memory usage
##
## memory and stack allocation
##
on tile 0 partition 1 has 64K memory and 4K stack
on tile 0 partition 2 has 32K memory and 4K stack
#on tile 0 partition 3 has 32K memory and 4K stack
on tile 1 partition 1 has 64K memory and 4K stack
on tile 1 partition 2 has 32K memory and 4K stack
#on tile 1 partition 3 has 32K memory and 4K stack
#on tile 2 partition 1 has 64K memory and 4K stack
#on tile 2 partition 2 has 32K memory and 4K stack
#on tile 2 partition 3 has 32K memory and 4K stack
##
## scheduling
##
## - max 3 slots per tile
## - it is allowed to not schedule anything on a tile
## - a partition can have more than one slot
## - the system partition always get a slot of 5000 cycles
##   at the end: you don't have to add this
##
#on tile 0 next slot is for partition 1 with 200000 cycles
on tile 0 next slot is for partition 1 with 1200000 cycles
on tile 0 next slot is for partition 2 with 10000 cycles
#on tile 0 next slot is for partition 3 with 377000 cycles
on tile 1 next slot is for partition 1 with 1200000 cycles
on tile 1 next slot is for partition 2 with 389000 cycles
#on tile 1 next slot is for partition 3 with 372000 cycles
#on tile 2 next slot is for partition 1 with 700000 cycles
#on tile 2 next slot is for partition 2 with 289000 cycles
# on tile 2 next slot is for partition 2 with 10000 cycles
# on tile 2 next slot is for partition 3 with 100000 cycles
```

- once all the above steps are completed then, open the two new terminal windows, connect the PYNQ board, and use the following command:

- Terminal 1 :
 - cd tutorial
 - cd monitoring-tools/
 - sudo ./channel_cheap_bridge 0 1 9878
- Terminal 2 :
 - cd tutorial
 - ./readout.sh

```

computation@computation-virtual-machine: ~
computation@computation-virtual-machine: ~
computation@computation-virtual-machine:~$ ssh student@pato-board.local
ssh: connect to host pato-board.local port 22: Connection refused
computation@computation-virtual-machine:~$ ssh student@pato-board.local
student@pato-board.local's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.14.0-xilinx-00014-g14b3cc2178b6 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Feb 20 21:17:04 2023 from 10.42.0.1
student@pato-board:~$ cd tutorial
student@pato-board:~/tutorial$ cd monitoring-tools/
student@pato-board:~/tutorial/monitoring-tools$ sudo ./channel_cheap_bridge 0 1 9878
+Socket created
+bind on port: 9878 done
+ Listen done
+Waiting for incoming connections.

```

- run the **main.m** script again, and then run the **SIL_Simulation_Parallel_Implementation.slx** simulation model. For every new execution of the Simulink model, make sure to run the main.m script before running the Simulink model; otherwise, it will give an error related to array dimensions.
- once the simulation is finished, it will generate the profiling report. In the report, the maximum execution time is the WCET of the particular task. Check the highlighted part from the following screenshot.

Code Execution Profiling Report for SIL_Simulation_Parallel_Implementation/Subsystem3

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See [Code Execution Profiling](#) for more information.

1. Summary

Total time	1278494475
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0F');
Timer frequency (ticks per second)	4e+07
Profiling data created	21-Feb-2023 15:51:55

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
Subsystem_initialize	1050	1050	1050	1050	1
(+) Subsystem_step [0.005_0]	14359225	14049378	1900	1900	91

3. Definitions

Execution Time: Time between start and end of code section.

Self Time: Execution time, excluding time in child sections.

- repeat the above steps to find the WCET of each application task. To understand the entire process, explanations are given considering the sensing task. Students should follow the same steps for finding the WCET of tasks such as sensing task 2, merging, and controller task. No need to find the timing for the actuation task.

Note: Find the WCET of each task individually; it is impossible to find them all together. So follow the below steps to avoid errors:

- 1) Generate the PIL block of a particular task for which WCET needs to find. Check the steps mentioned above.
- 2) Keep the original block, i.e., (S-function block aside as commented.) If it is not commented, then both the PIL and original blocks will run together

and give the run-time error.

- 3) Other blocks should be as it is in the entire Simulink model.
- 4) Note down the timing as explained above.
- 5) Repeat the same step for another task.