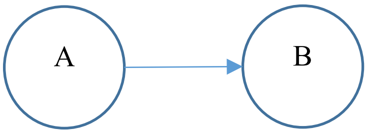
Tutorial: Inchron Tool Suite

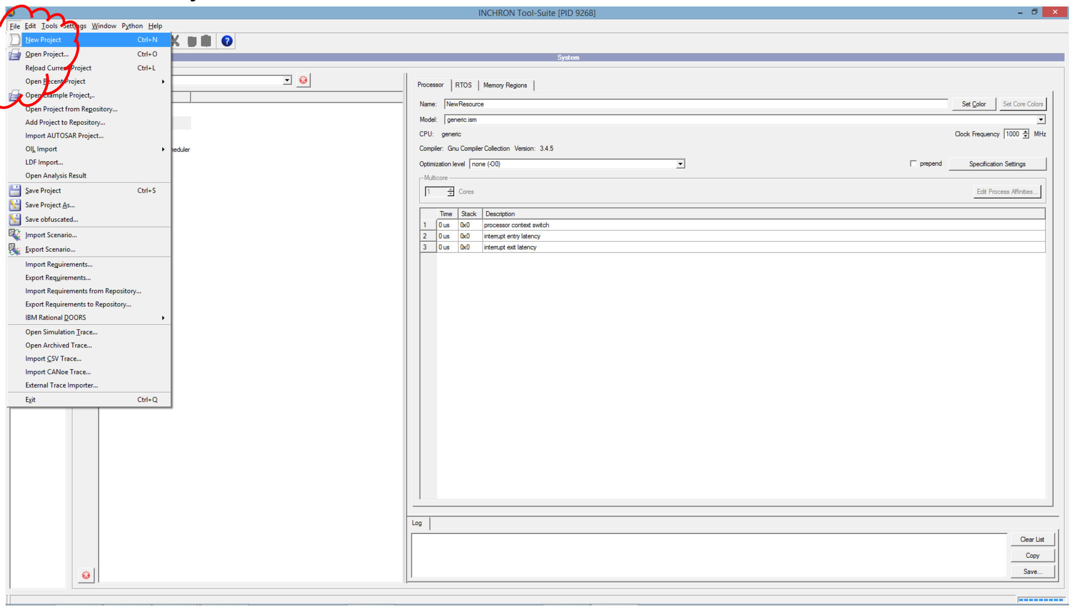
Contents

1. Modelling a simple task graph

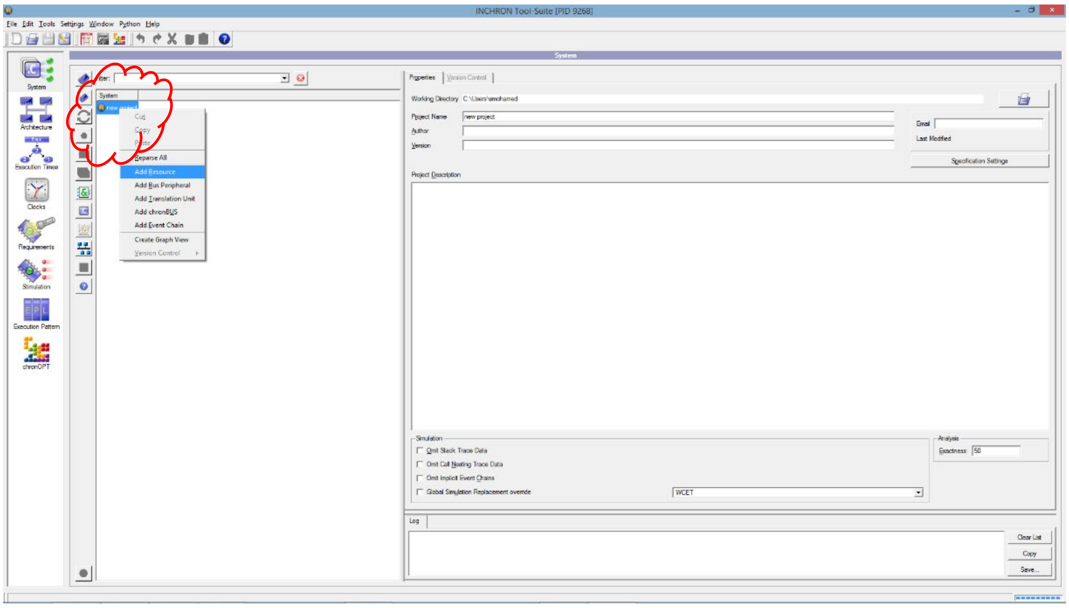


Case 1: Task A and Task B are mapped to Processor P1. Task A arrives periodically with a period of 40ms. Task B executes after A. Task A takes 10 ms and Task B takes 20 ms to execute.

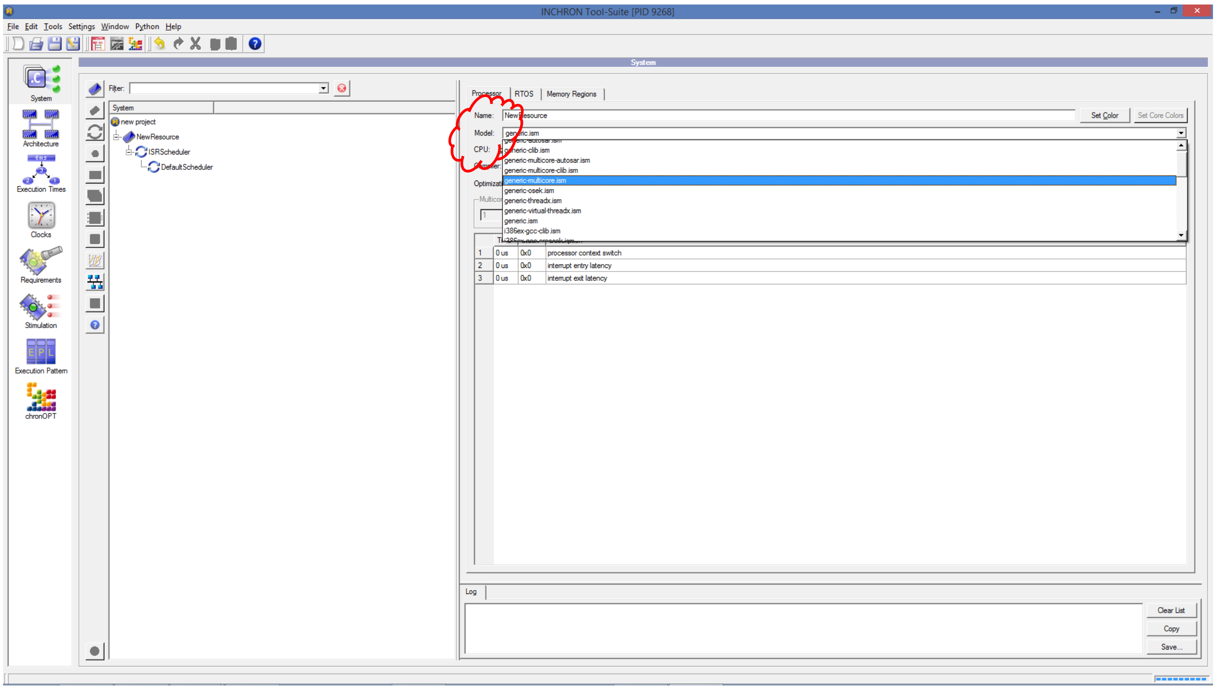
1. Start Inchron Tool-suite and create a new project (Ctrl + N).
   File → New Project



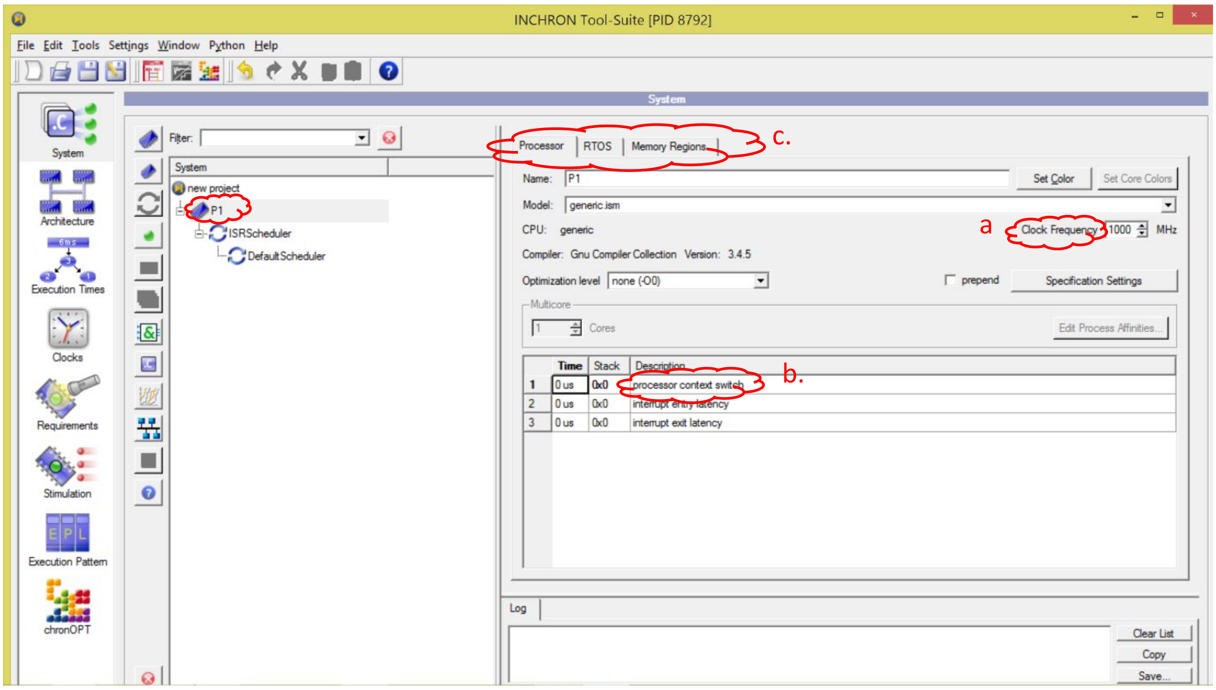
2. Add resource or Processor P1.
    new project → Add Resource



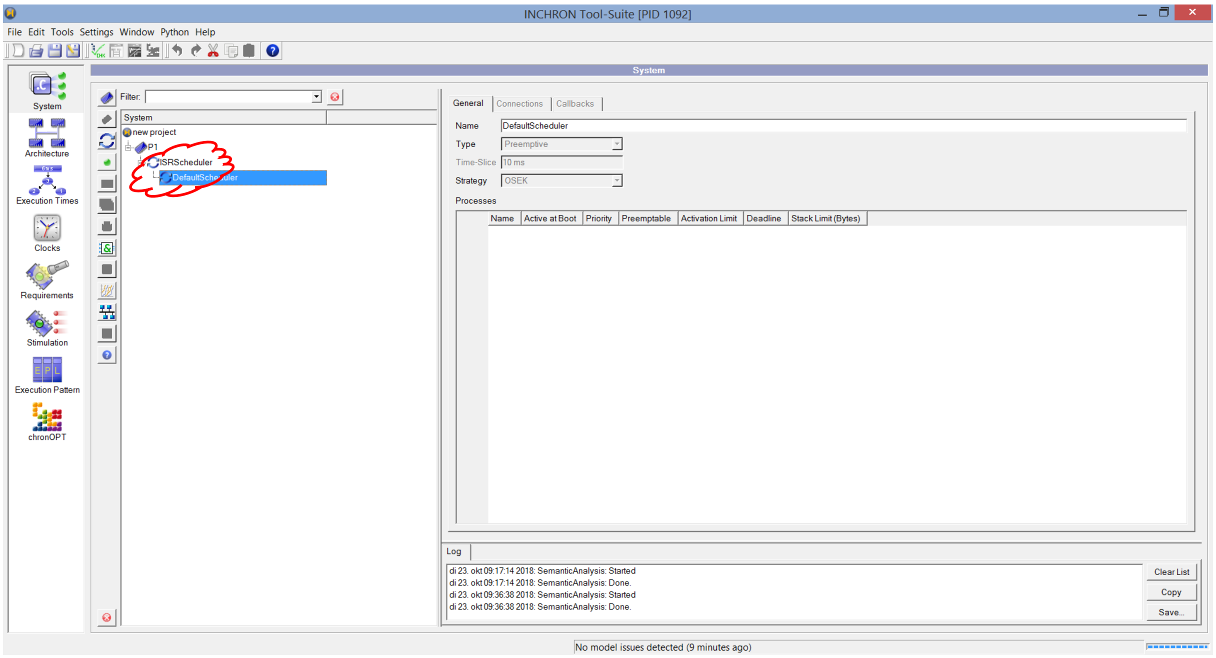
By default, this adds a ‘generic.ism’ processor. You can rename the ‘NewResource’ to P1.
To choose a different type of processor, for instance, a multicore processor:
Choose from the drop-down menu for ‘Model:’ → ‘generic-multicore.ism’



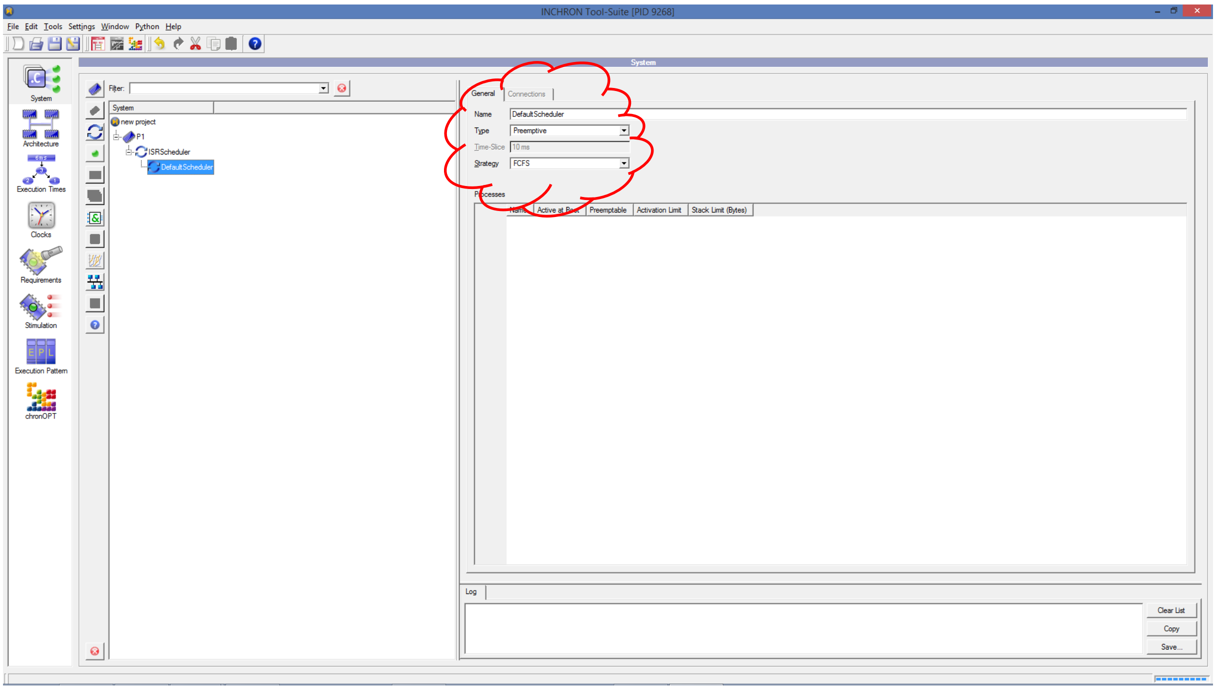
For this tutorial, we will use generic-osek.ism, because this is also the processor that will be used in the rest of the project.
Choosing a processor comes with a lot of pre-set options. You could choose (a.) clock frequency for your processor, (b.) context switch, if any and other options (c.) available on the tabs. Only use if these options are relevant or else leave it as default.



3. To choose the scheduling strategy for the tasks running on the processor,
Click on 'Default Scheduler'



You can then see the 'Type' and 'Strategy' on the right side. By default it is Preemptive, OSEK.
This cannot be changed as all processors used in this project are OSEK processors.
For this tutorial, we will choose the default option.

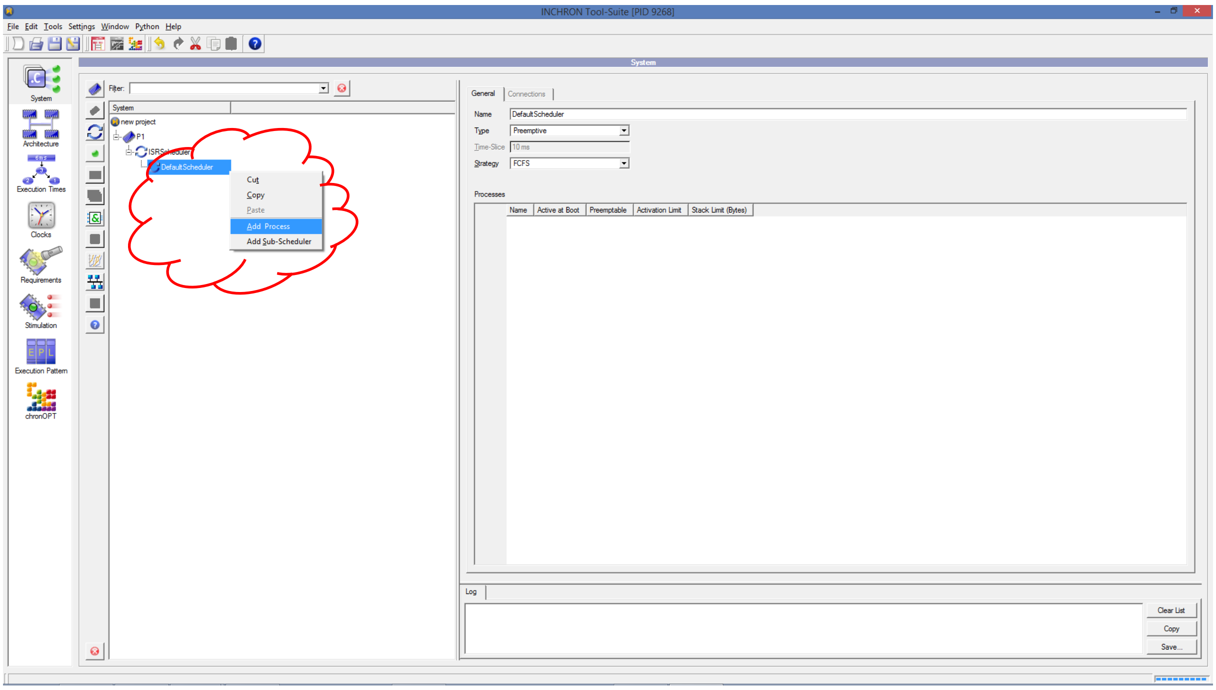


4. Adding Tasks.

Default Scheduler → Add Process

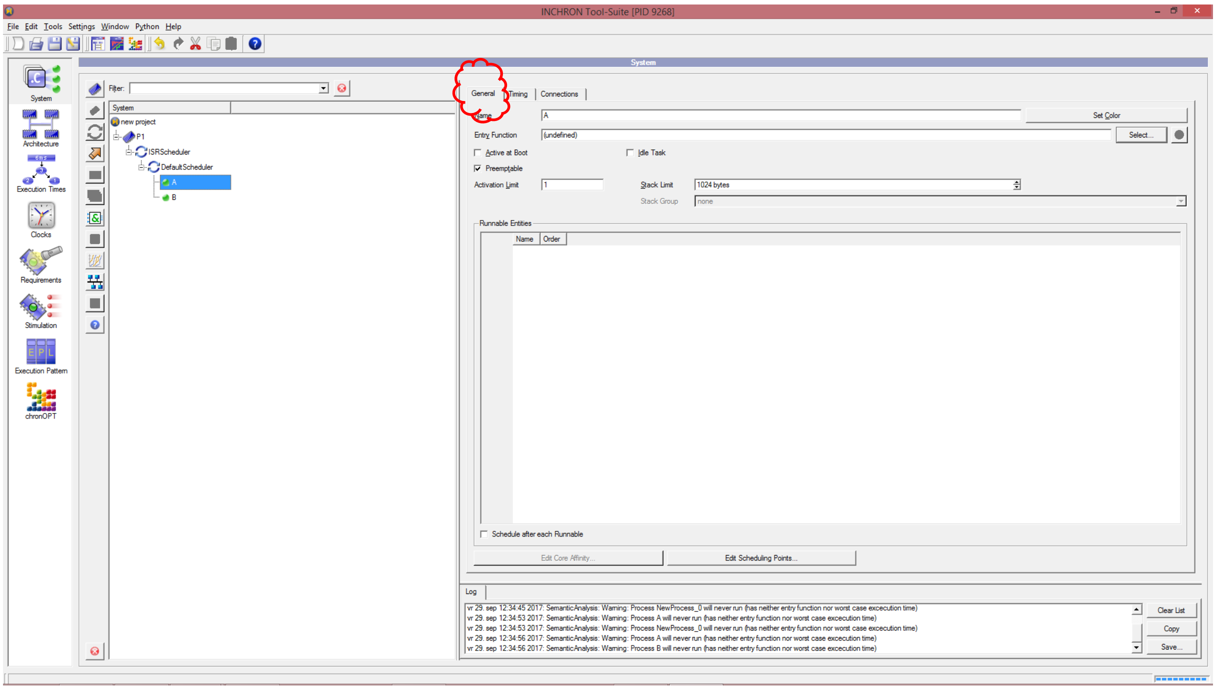
Default Scheduler → Add Process

Note that you get an interrupt instead of a process if you click on "Add Process" after right-clicking "ISRScheduler"

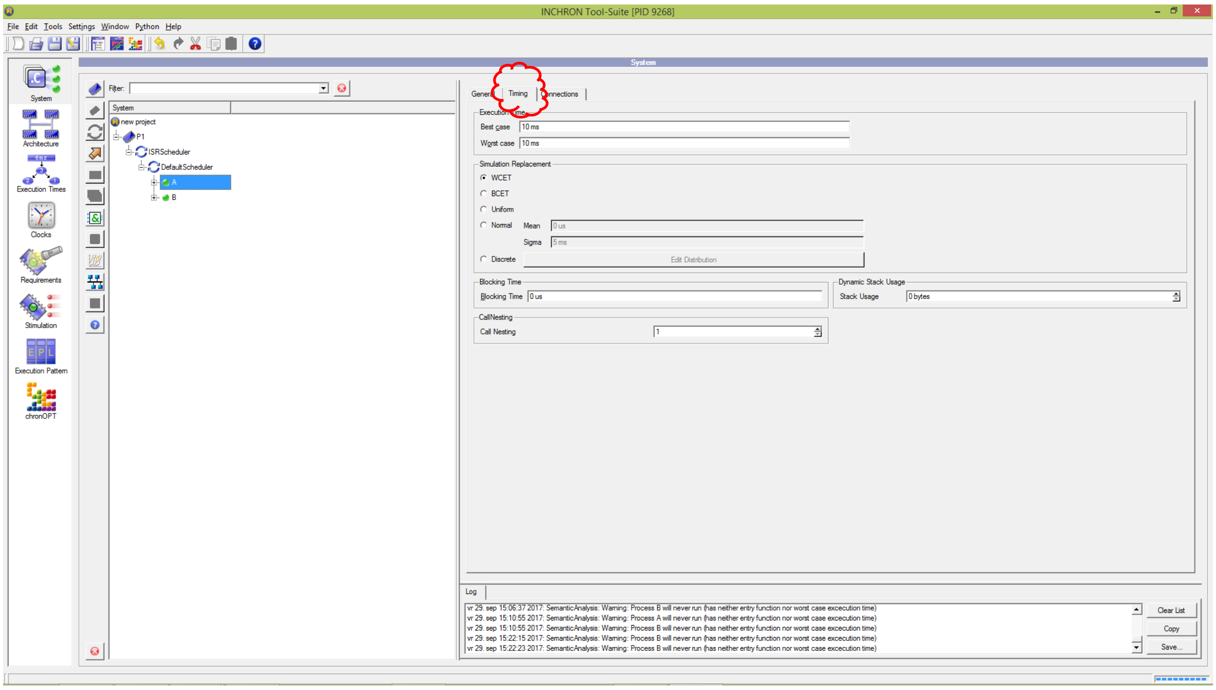


A default 'NewProcess' would be created. A Process simulates a task. You could rename the processes to A and B. For each Process you have three tabs – General, Timing and Connections. In General tab, you could choose the following checkboxes:

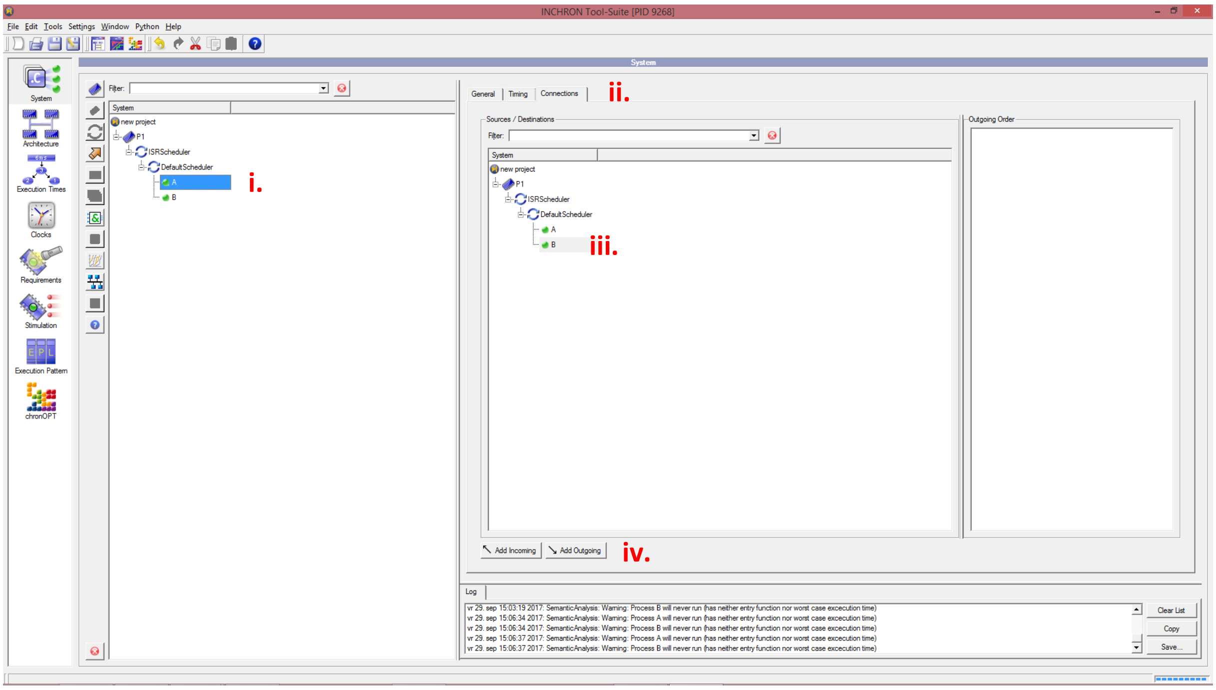
a. Activate at boot (unchecked for this tutorial)
b. Preemptable (checked)



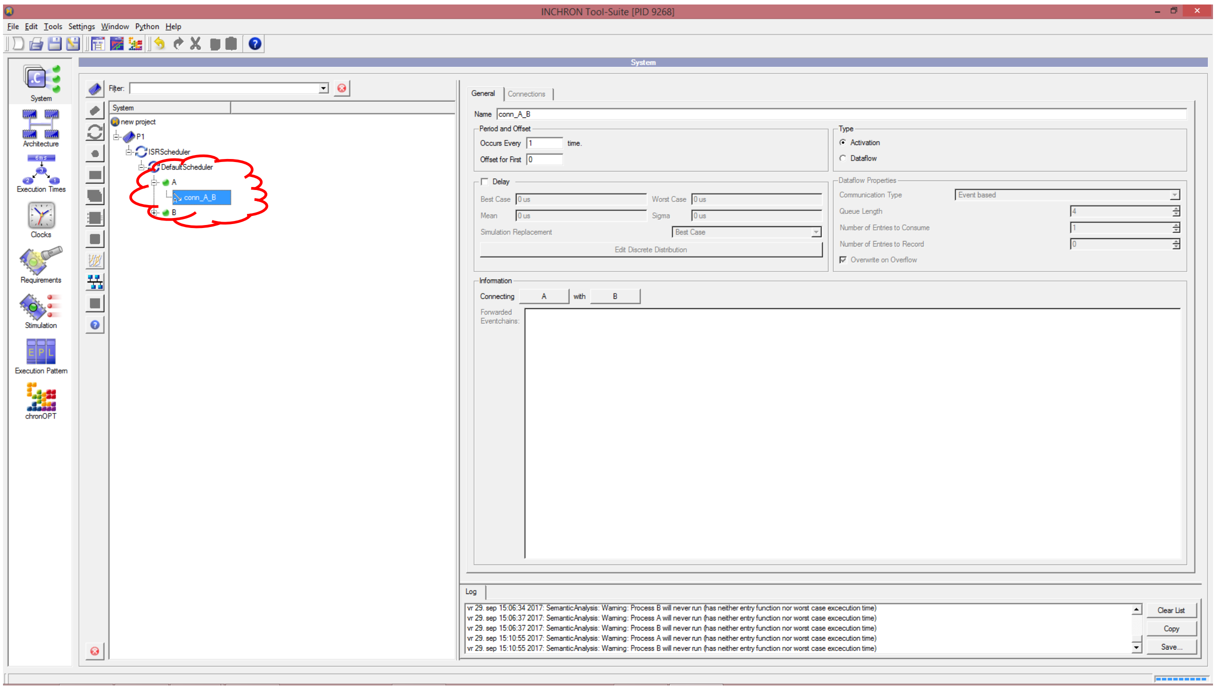
In Timing tab, you could set best case and worst case execution time for the task. For this case we give 'Best case' and 'Worst case' execution times of A to be 10 ms and for B to be 20 ms respectively.



5. To add dependency from A to B,
   i. choose Process A
   ii. choose Connections tab
   iii. choose Process B
   iv. choose 'Add Outgoing'



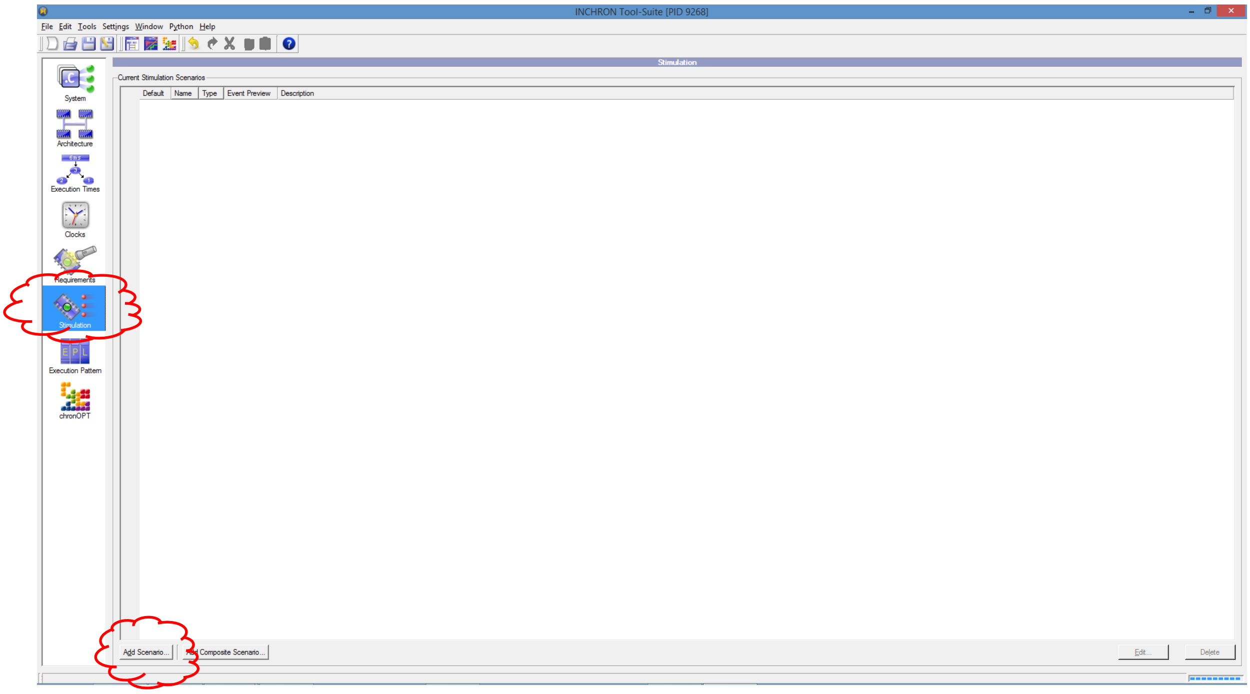
The dependency edge from A to B would be automatically added with default name 'conn_A_B'. By default this connection is set as an activation connection. There are several options for this connections (like a delay, period, etc.)



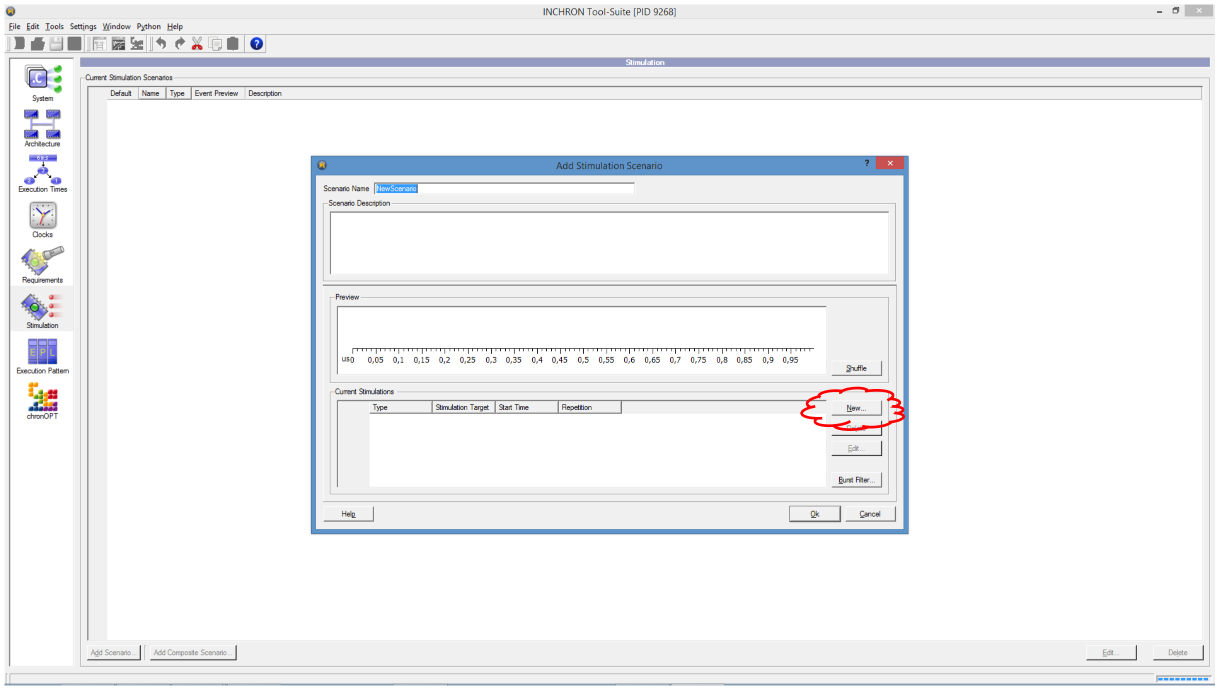
6. Simulation

To simulate the task graph, we need to define the arrival sequence for tasks. In our case, we have Task A with period 40 ms. To model this –

i. Choose ‘Stimulation’ → ‘Add Scenario’



ii. In the ‘Add Stimulation Scenario’ window, choose ‘New’

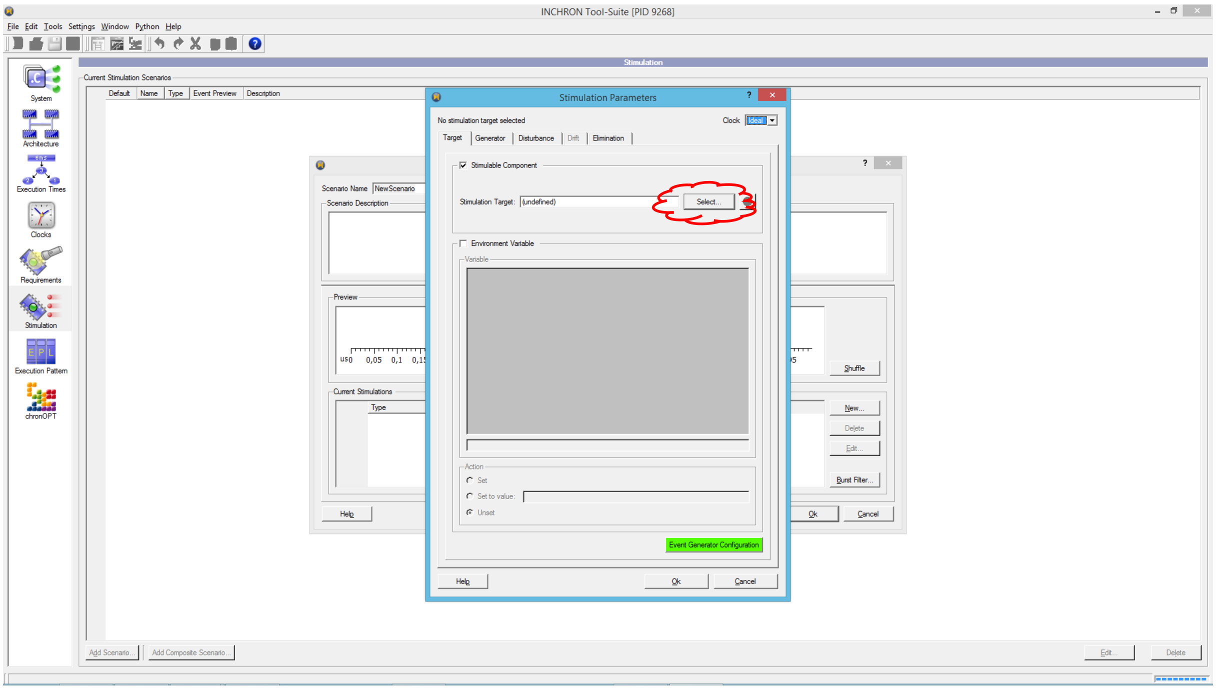


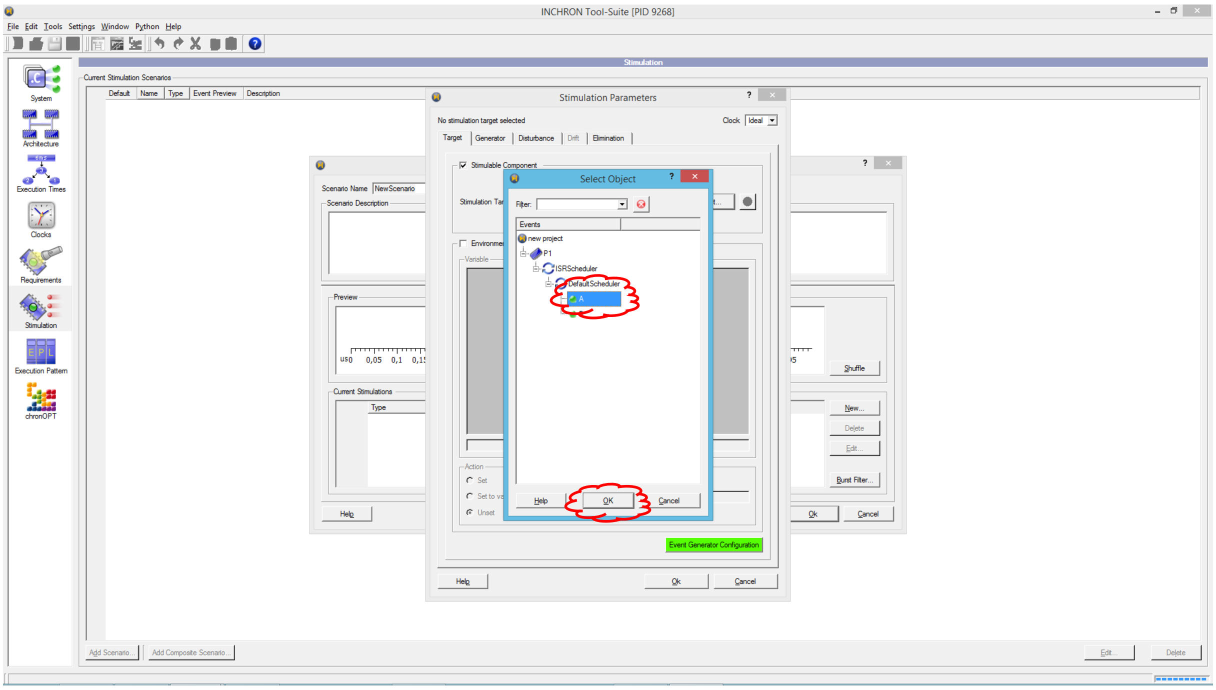
A ‘Stimulation Parameters’ window would open up. This window has multiple tabs. We are only concerned with ‘Target’ and ‘Generator’ tabs.

iii. In the ‘Target’ tab, choose the stimulation target by clicking on ‘Select’.

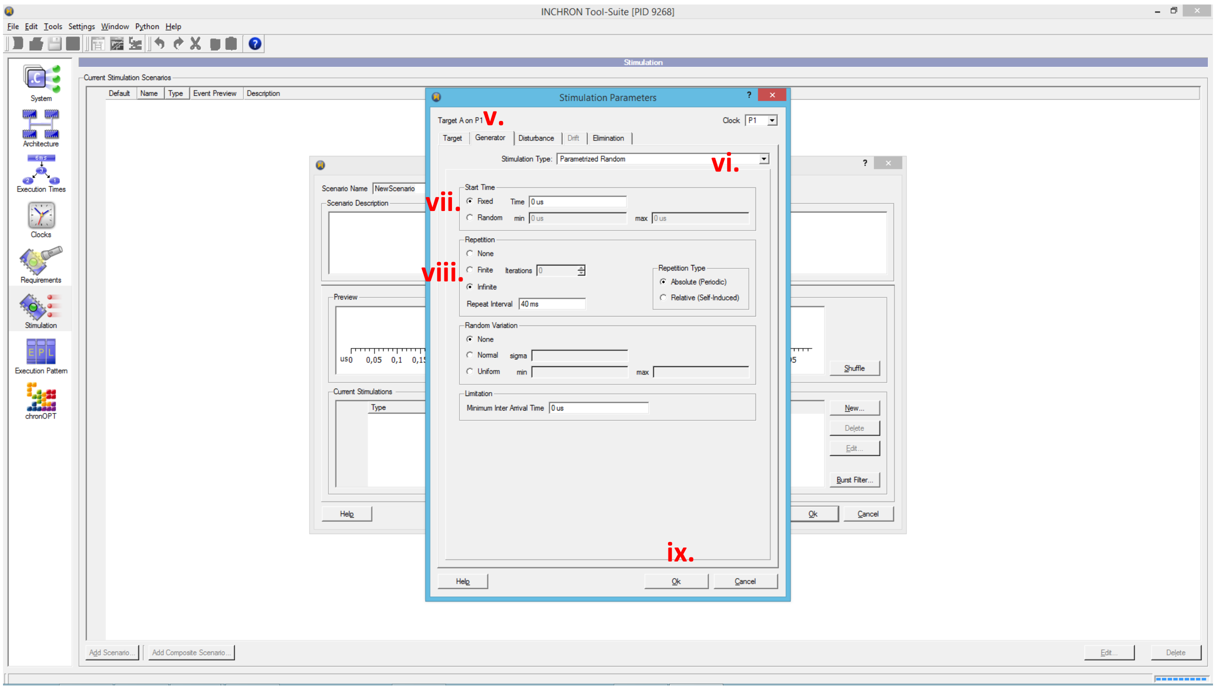
A new window asking to ‘Select Object’ would pop-up.

iv. Choose A (task to stimulate) and click ‘OK’.

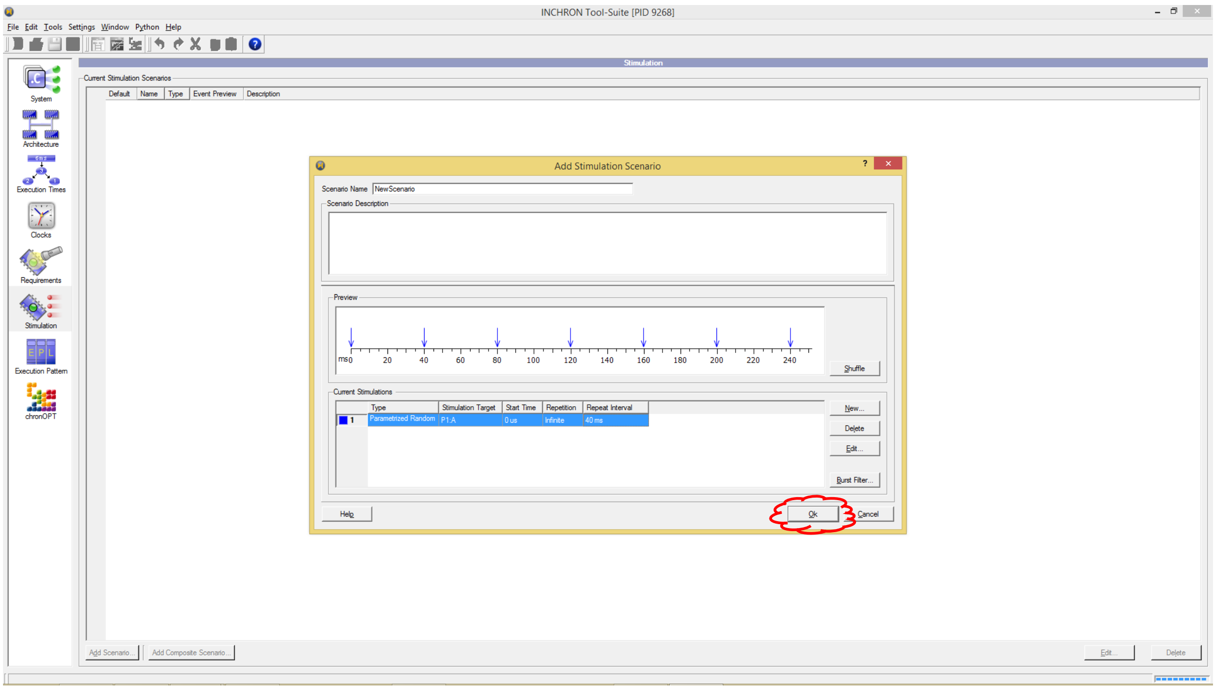




v. Go to the ‘Generator’ tab.
vi. Choose the stimulation type to be ‘Parameterised Random’.
vii. If the ‘Start Time’ of task is not at 0 s, choose the appropriate start time. For this case, the start time is 0 s.
viii. If the task is periodic, in the ‘Repetition’ section, choose appropriate options. For this case, we consider infinite repetitions with a period of 40 ms.
ix. After choosing the required options, click ‘OK’



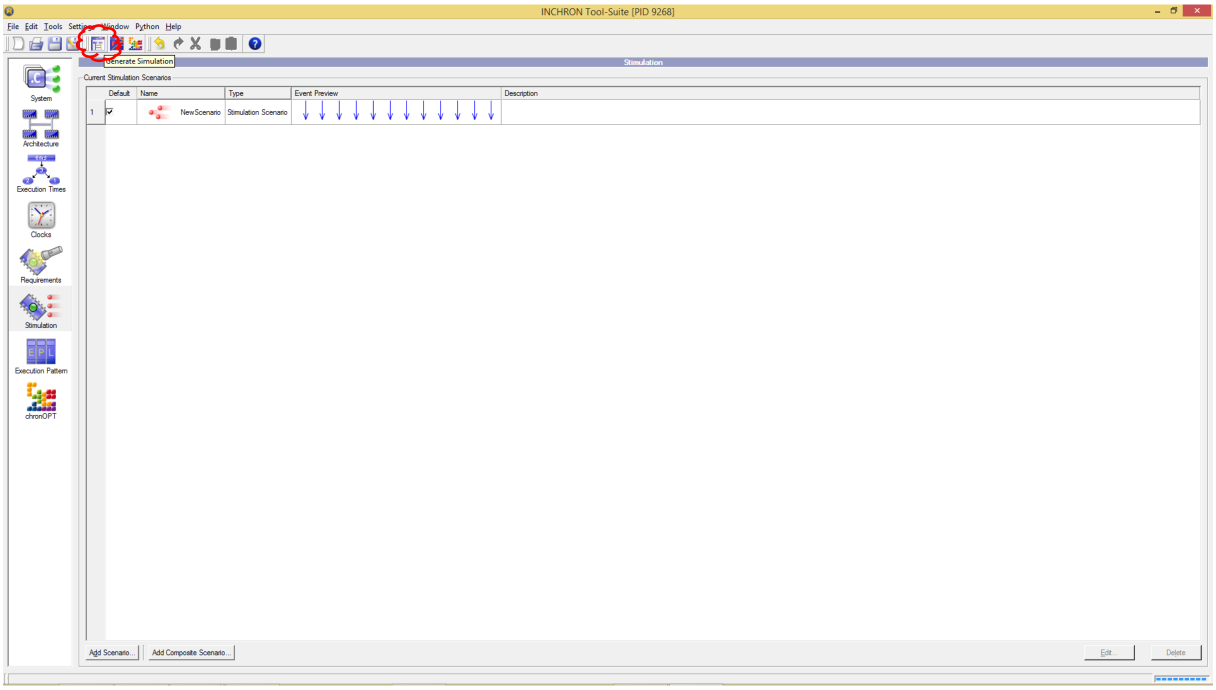
x. You will now see the following screen. To add more tasks (if needed), follow steps (ii) to (ix). Else, click 'Ok'.



xi. Now, you would see a 'NewScenario' added in the Stimulation. We can now directly go to generating simulation.

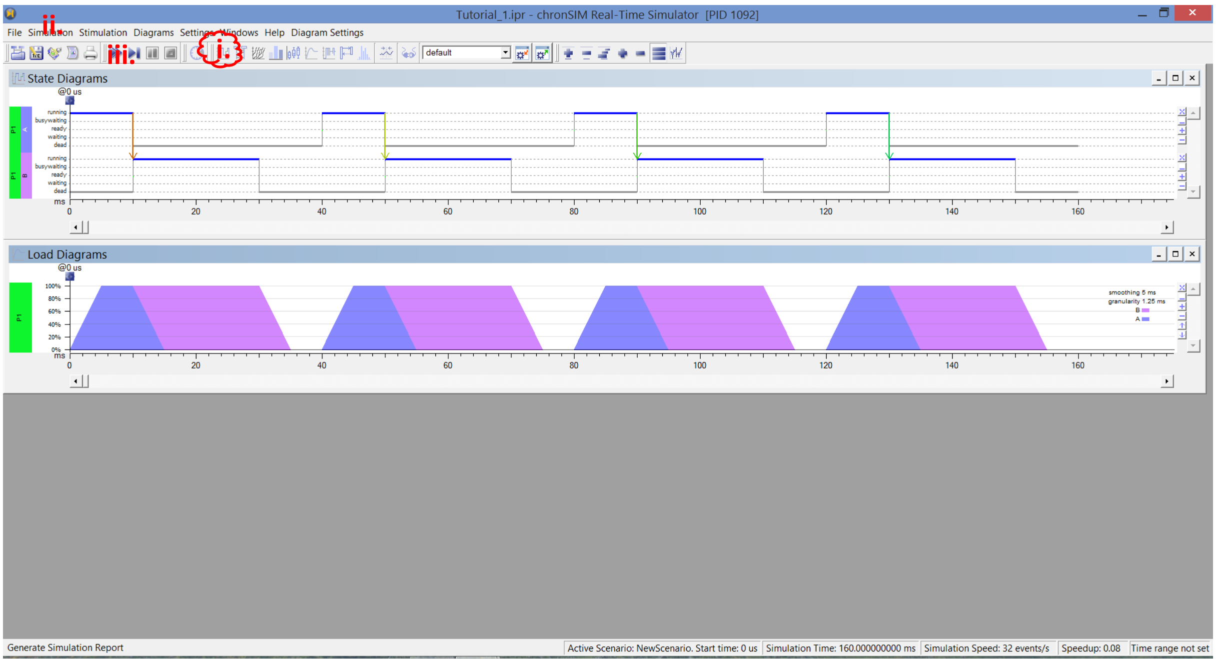
7. Generate Simulation

Use one of these options. Tools → Generate Simulation or Ctrl + R or click on the icon.



A new 'chronSIM Real-Time Simulator' window would open.

i. If you see a blank screen, click on 'Show state diagrams' or choose 'Diagrams → Show State Diagrams'. Alternatively, you could also explore other visual diagrams that you seem fit from the 'Diagrams' tool bar. One of these diagrams is the load diagram. This is also a useful tool to see whether there is still space on a processor for more tasks.



ii. You could also set the simulation to run for a specified time or set breakpoints using 'Windows → Show Simulation Control'. You may also use other simulation options.

iii. You can now run the simulation.

a. 'Simulation → Run Simulation' Ctrl + R

b. 'Simulation → Step into Simulation' Ctrl + I

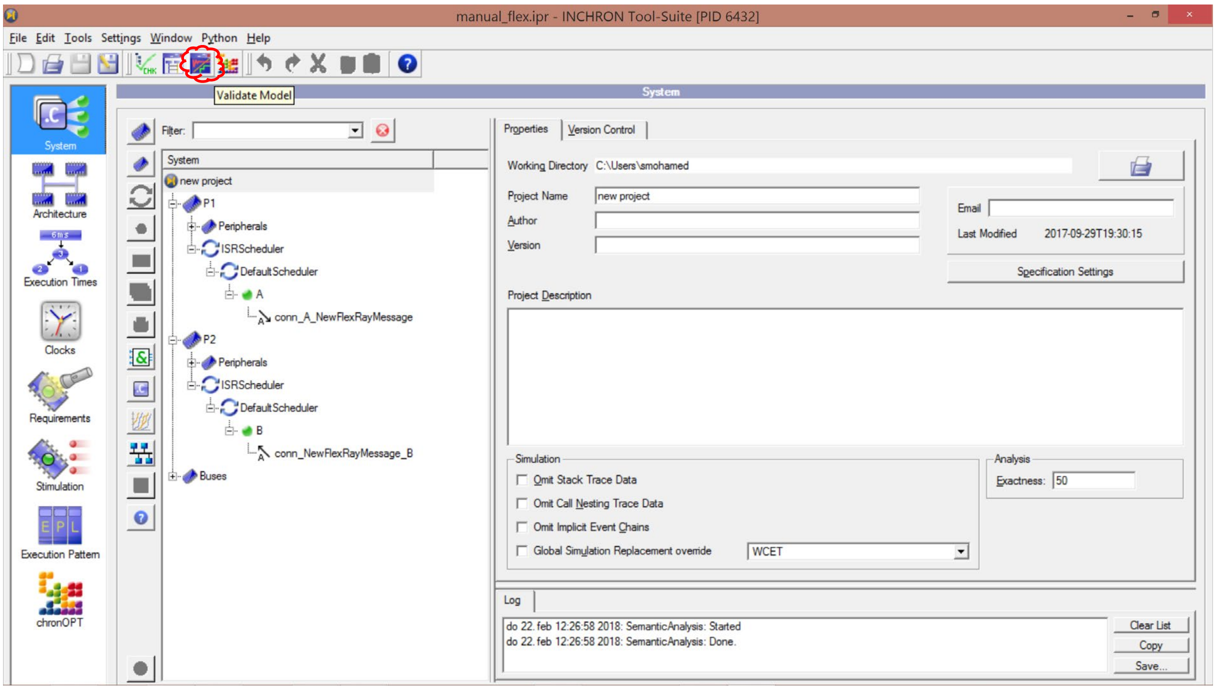
c. 'Simulation → Stop Simulation' Ctrl + Q

You may use the corresponding shortcut icons.

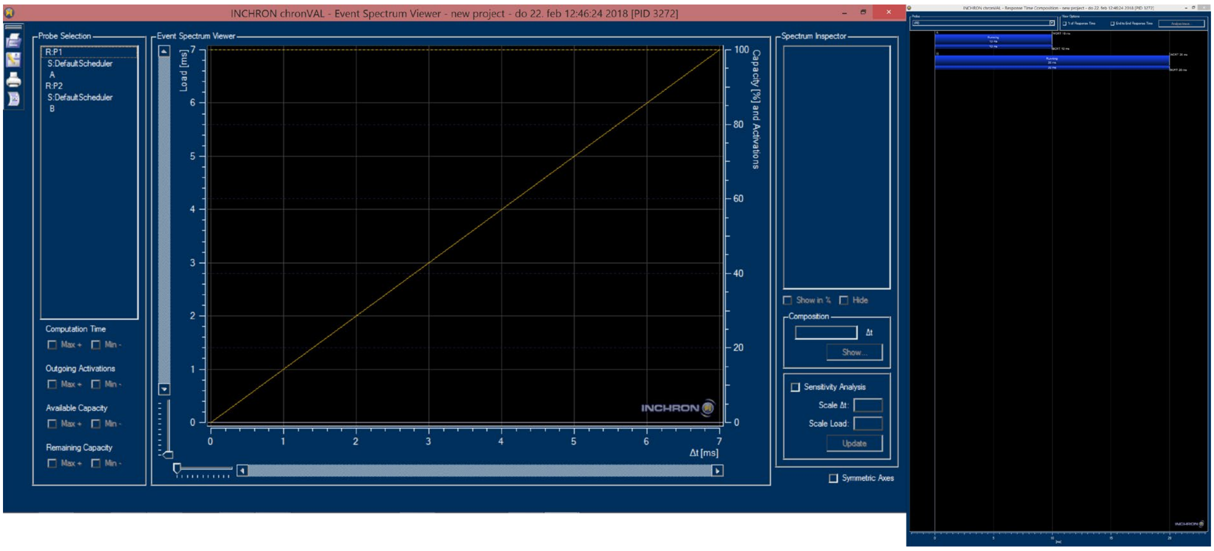
NOTE: If Inchron simulation is running, it will eat up your memory. It is suggested that you stop the simulation after sufficient time.

8. Validate Model (chronVAL)

Use one of these options. Tools → Validate Model or Ctrl + R or click on the icon.



Two new windows will open – INCHRON chronVAL 'Response Time Composition' and 'Event Spectrum Viewer'. 'Event Spectrum Viewer' is not considered in the scope of this tutorial. To understand more about 'Event Spectrum Viewer' refer to Real-Time Calculus and Approximative Real-Time Analysis.



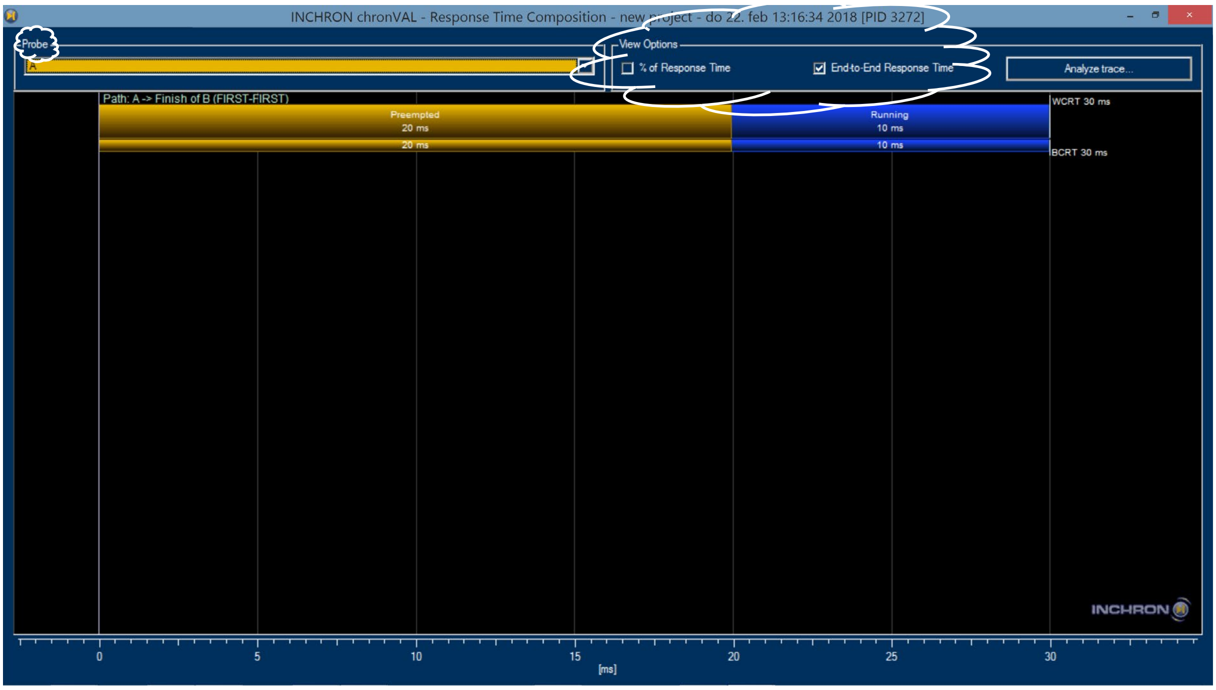
We only consider Response Time Composition window in this tutorial. In this window, you have

i. Probe – That can be used to analyse tasks, messages and event chain individually. (default: (All), you can use the drop-down menu to choose others.)

ii. View Options – By default (uncheck '% Response Time' and 'End-to-End Response Time'), the response time analysis shows worst-case response time (WCRT) and best-case response time (BCRT) values in absolute time domain. (Refer 'Manual' for more details)

   a. 'End-to-End Response Time' option is checked to see response time of a path (you can define a path using 'Event Chain'. if not defined, Inchron detects one). When you check this option, note that defined 'event chains' are added to 'probe'.

   b. '% Response Time' option changes the x-axis to a percentage of total response time. This option is useful to evaluate the cases where you have pre-emption, unavailability of ECU and similar.

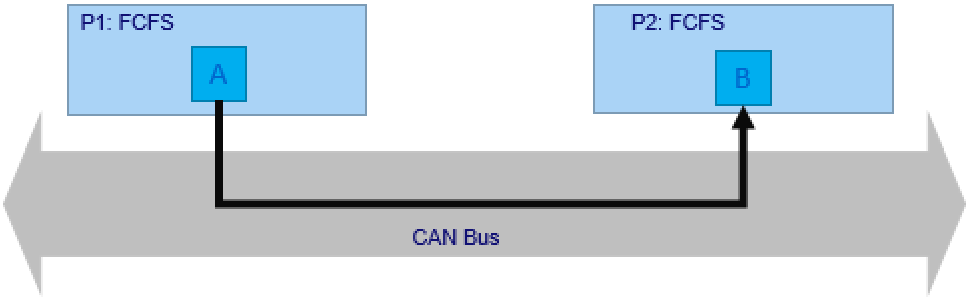


iii. 'Analyze Trace' button can be used to show the trace in chronSIM which yielded the result. This option becomes active for specific probes.

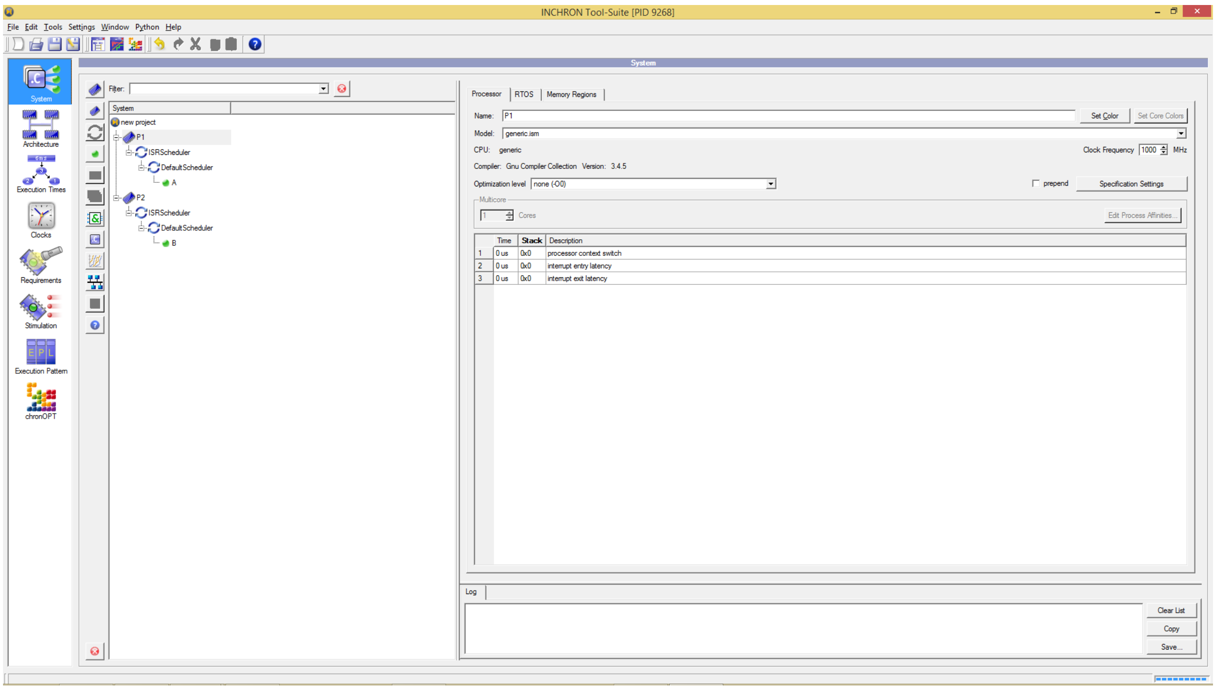
You see the following chronVIEW when you click on the ‘Analyze Trace’ button. In this view, you see the scenario which induced the BCRT and WCRT for the probe we considered.

2. Setting up a CAN bus

Let us consider the same task graph as in 1. The platform here is two processors connected by a CAN bus. Task A runs on P1 and Task B runs on P2. Task A sends message to Task B through a CAN bus.



1. Model the platform (two resources for processors and two tasks A and B) without CAN bus.

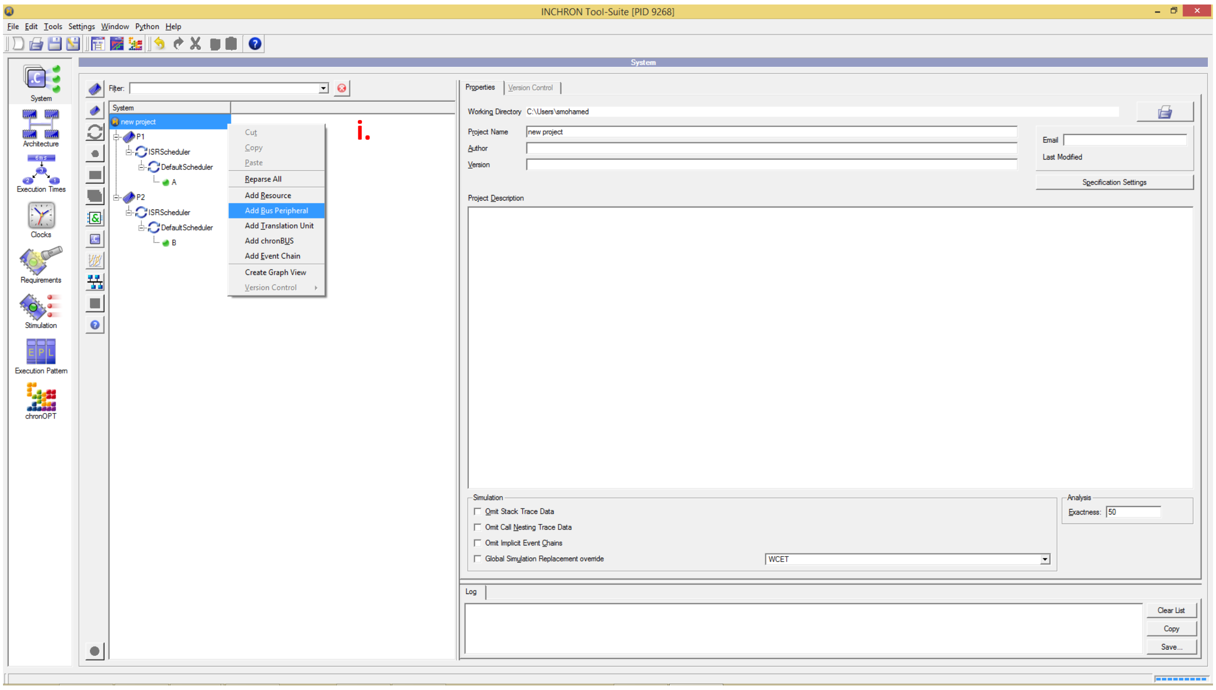


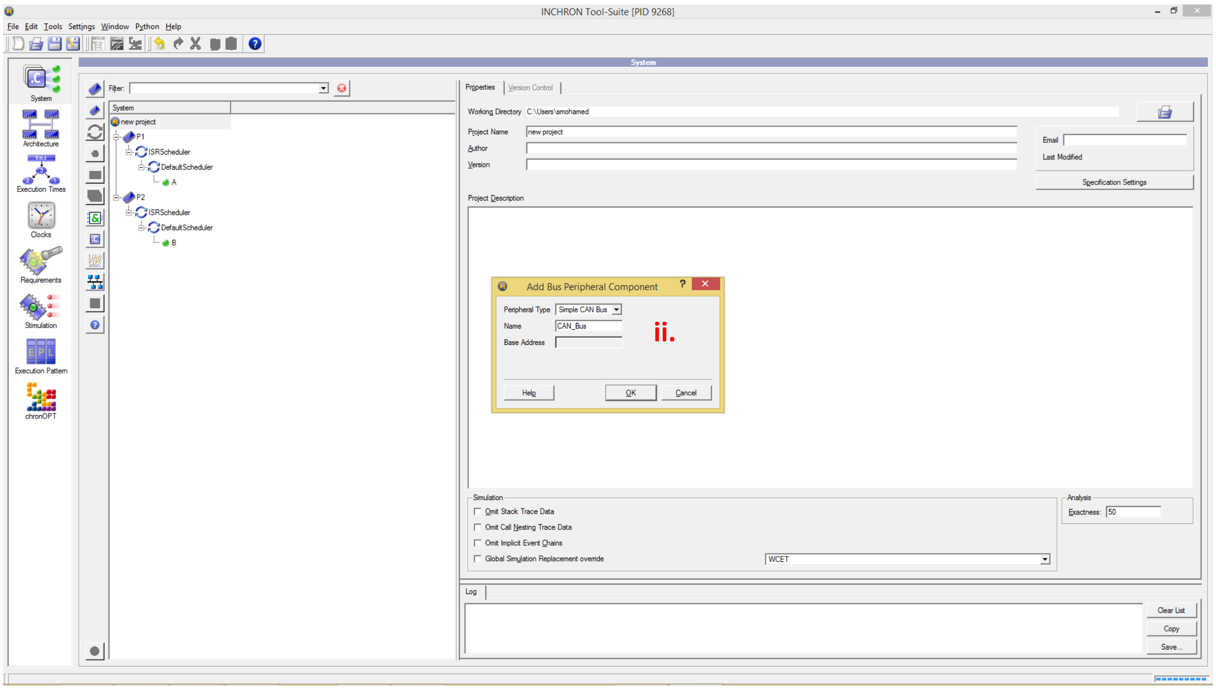
It would look like above. Now, we have to add a CAN bus that connects P1 and P2.

2. Initialising CAN bus

   i. new project → Add Bus Peripheral

   ii. In the 'Add Bus Peripheral Component' window, choose 'Simple CAN Bus' as the peripheral name. Give a name, CAN_Bus.





Now our platform has a CAN bus.

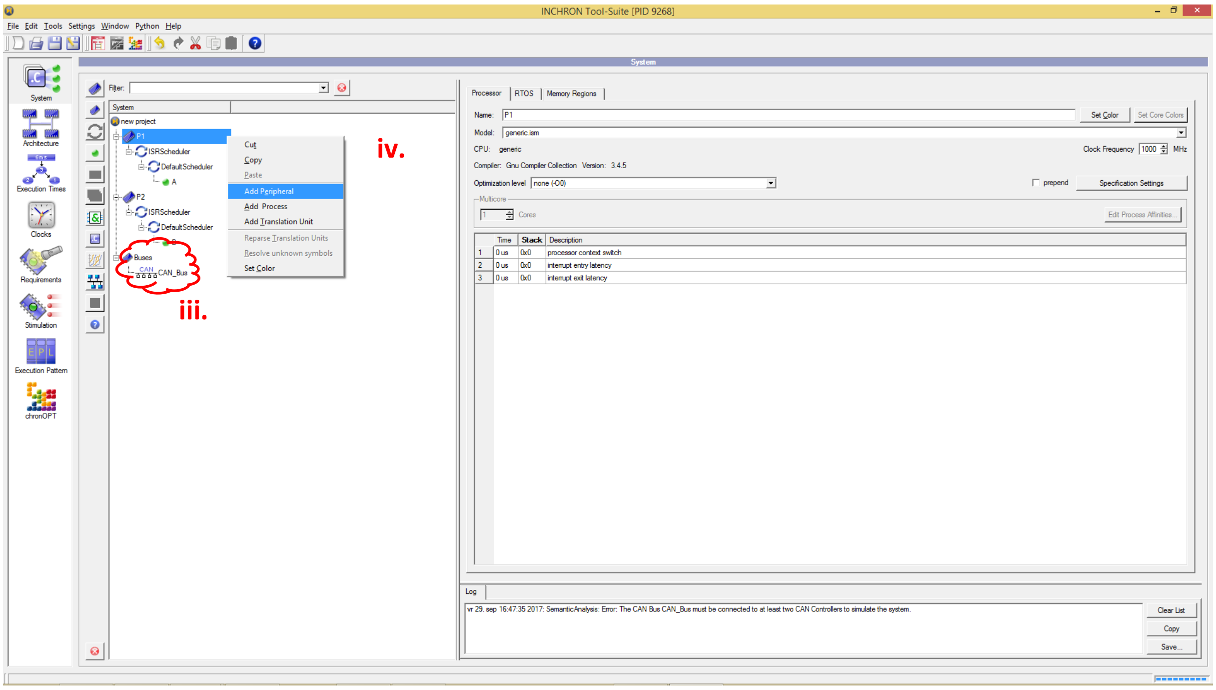
iii. Choose the ‘CAN_bus’ under ‘Buses’ and you will see a ‘Properties’ tab which lets us change the ‘baudrate’ of the CAN bus. Changing the ‘baudrate’ indirectly changes the BCET and WCET of the messages (based on message size)

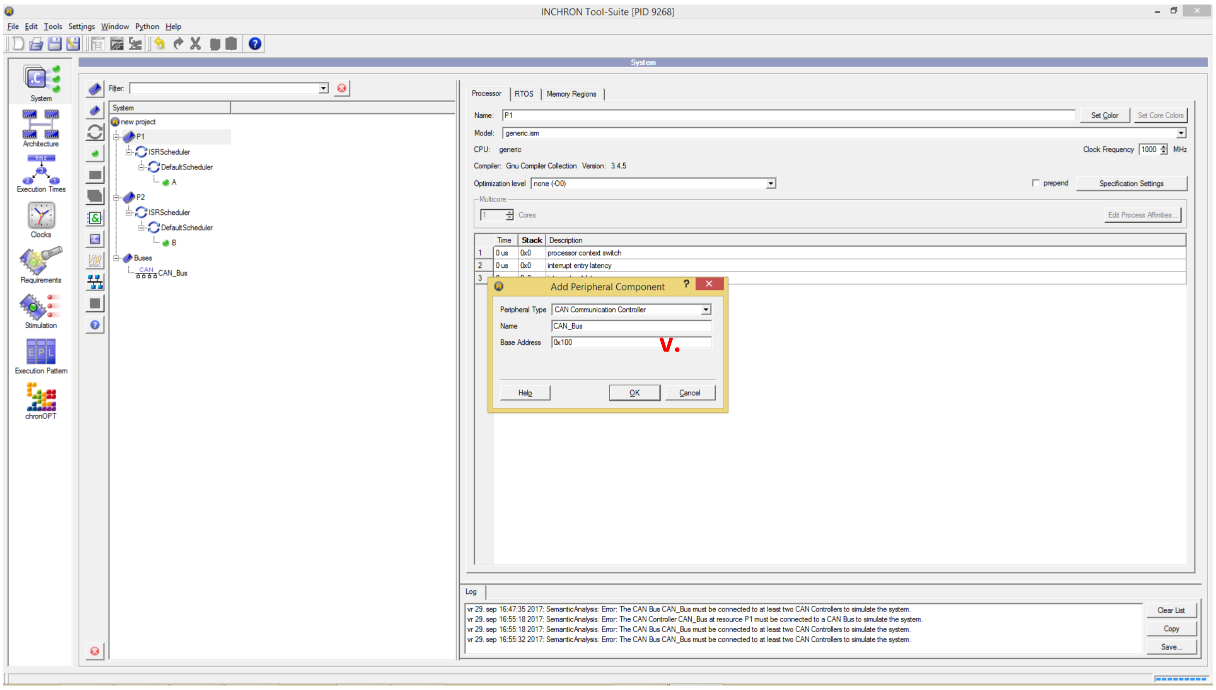
Now we need to connect our Processors to the CAN bus. For this we need to add a CAN bus peripheral to each Processor.

iv. 🖰 Resource (P1 or P2) → Add Peripheral

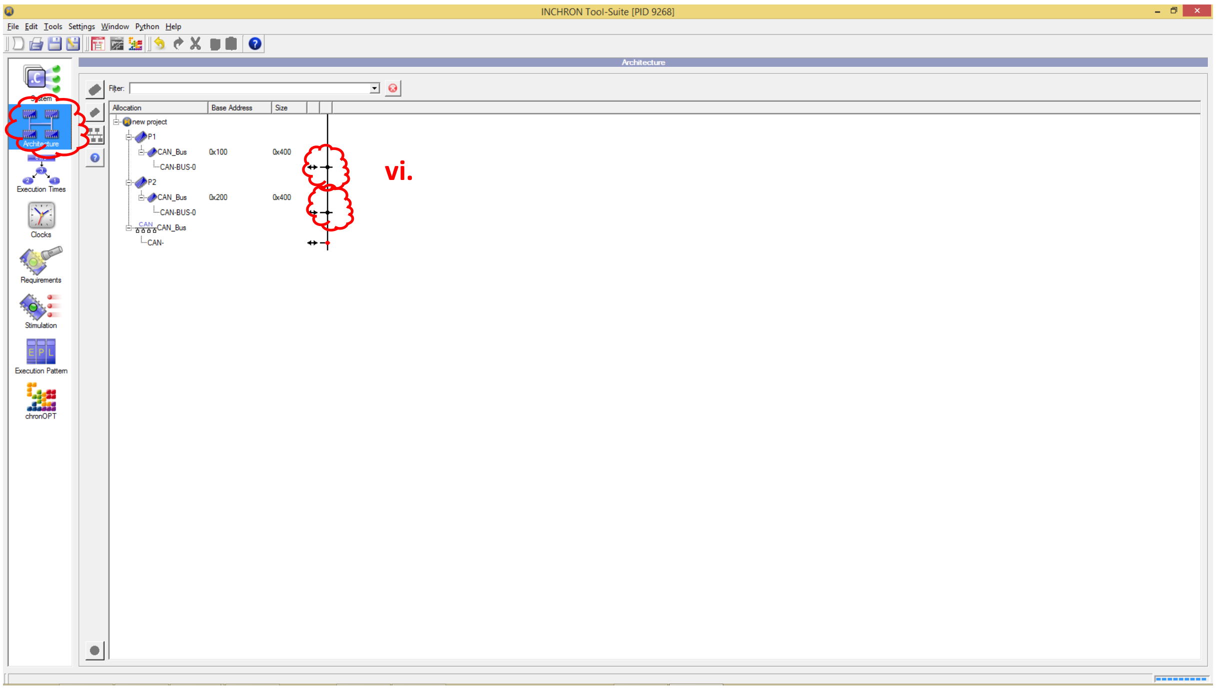
v. In the ‘Add Peripheral Component’ window, choose ‘CAN Communication Controller’ as ‘peripheral type’, a name (CAN_Bus) and a base address (0x100).

The above steps (iii – iv) need to be done for all processors connected to the CAN Bus.





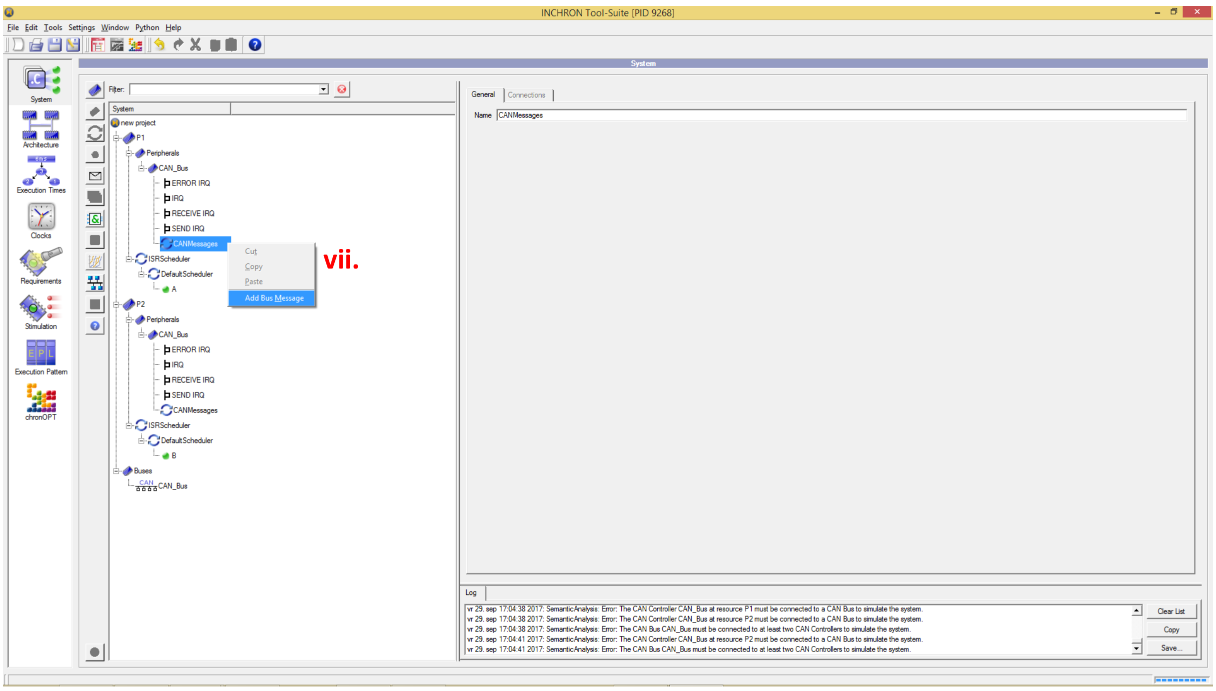
vi. Connect the ‘CAN communication controllers’ in P1 and P2 to the CAN bus in the ‘Architecture’ view.
Click on the graphical node between CAN-BUS-0 in P1 and P2 with the vertical line. (Highlighted)



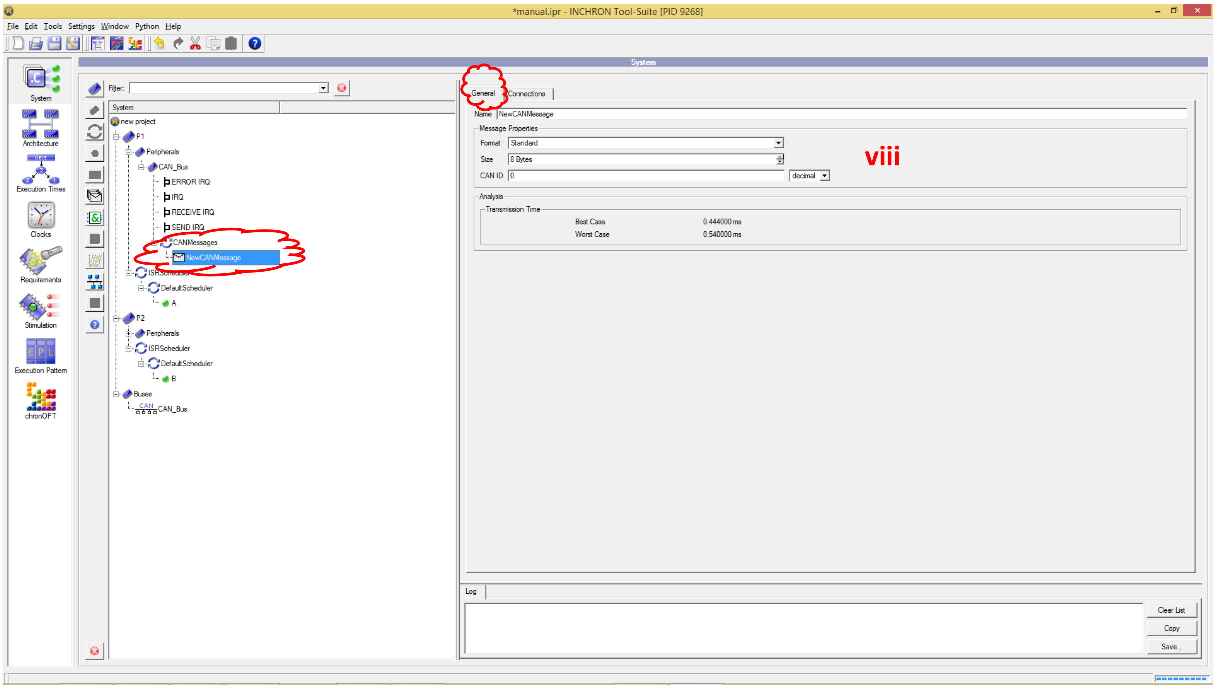
vii. Now we need to add the CAN bus messages. Go back to ‘System’ view.

🖱 CANMessages → Add Bus Message

NOTE: Even though we need ‘CAN Communication Controller’ for both the processors, we only need one ‘CANMessage’ at either P1 or P2 to communicate from A to B.

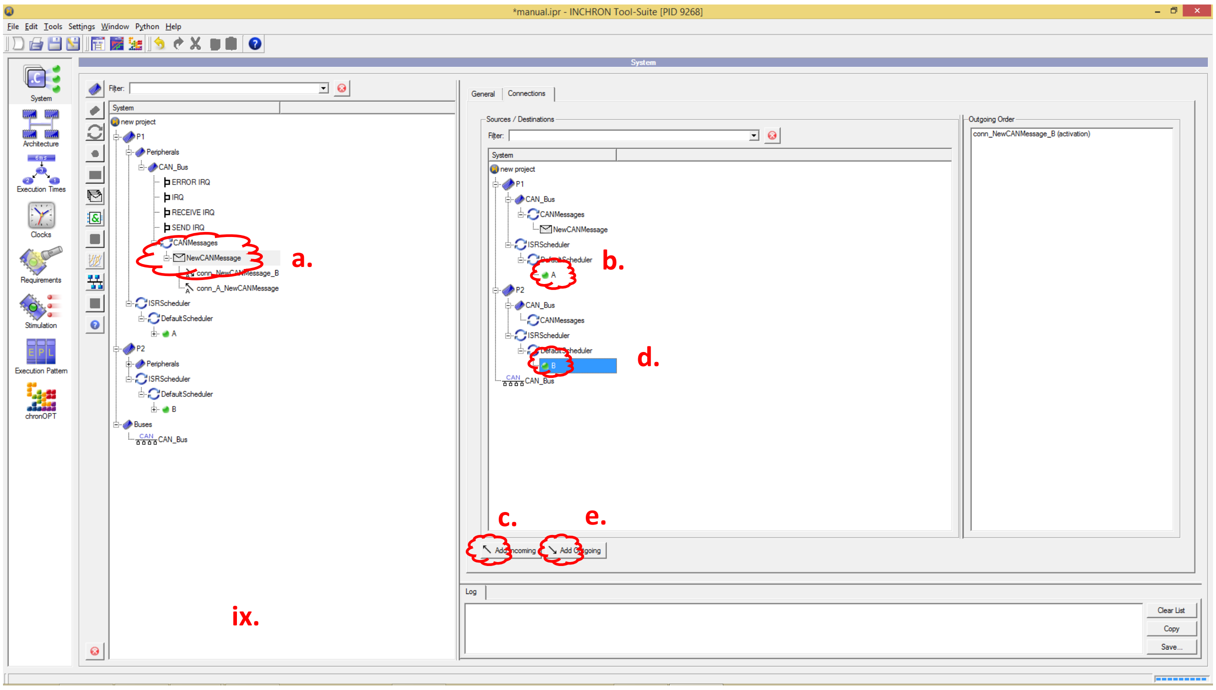


viii. A ‘NewCANMessage’ would be generated. Choosing this, we would see two tabs. A ‘General’ tab has options for ‘Name’ and ‘Message Properties’. In Message Properties, keep the format as ‘standard’; ‘size’ refers to the size of the message (max 8 Bytes, if message size is more, you need to split the message and add dependencies); ‘CAN ID’ refers to the message ID (needed for assigning priorities). Transmission time here is dependent on the size of the message and baud rate (see 2.2.iii to change it).

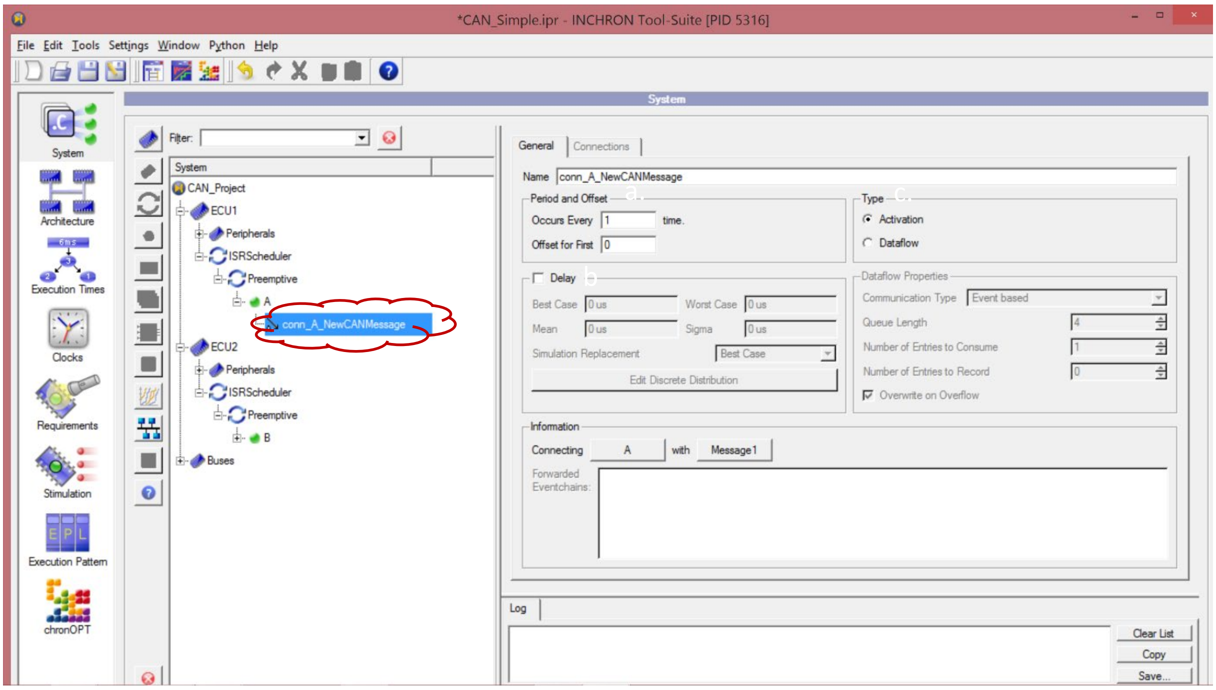


ix. We can now send message from A to B through this ‘NewCANMessage’. For this,

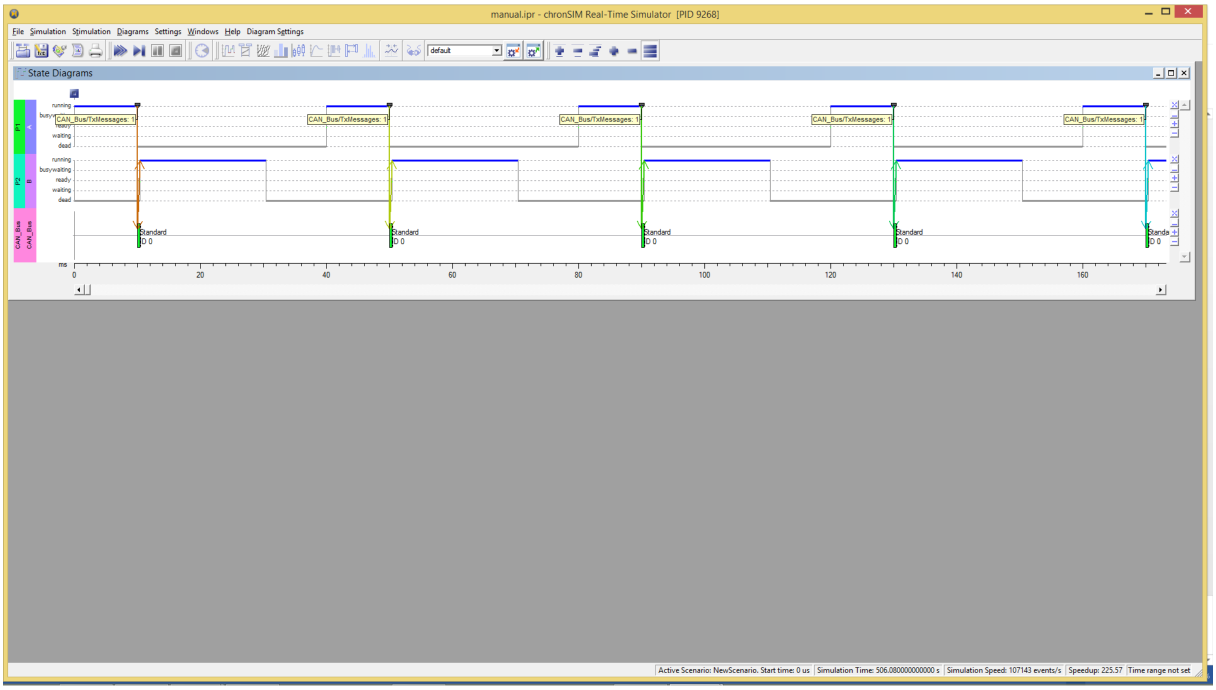
a. Choose ‘Connections’ tab of ‘NewCANMessage’.
b. Click on Process ‘A’ and
c. Choose ‘Add Incoming’.
d. Then, click on Process ‘B’ and
e. Choose ‘Add Outgoing’.



x. Choose the ‘conn_A_NewCANMessage’ (we could choose this from either the CANMessages or from Process A). We could edit (a.) period and offset of the message, (b.) add delay and (c.) make the message a dataflow or activation connection. For now, keep this as default.



3. After doing all the above steps (in 2.2), you can simulate as explained in 1.7. (NOTE: you should have a stimulation scenario as explained in 1.6).

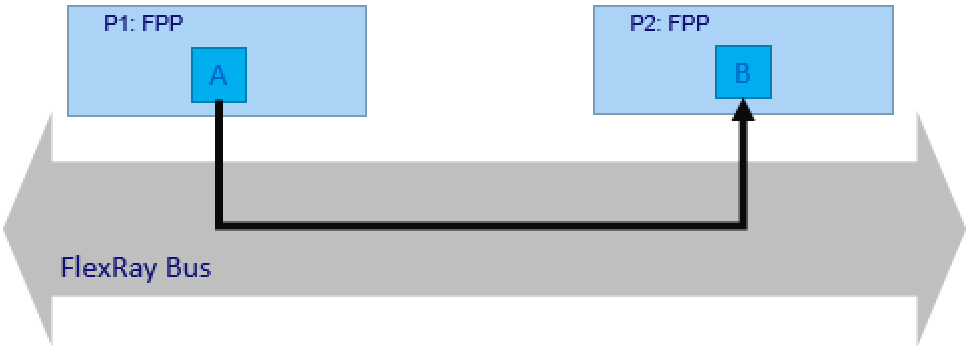


4. You can also validate the same model (explained in 1.8). The ‘Response Time Composition’ window with checked ‘% Response time option’ is given below.

3. Setting up a FlexRay bus

FlexRay is another communication bus similar to CAN. To setup FlexRay bus, you need to follow the same procedures as you set up CAN bus.

We will consider the same setup as in Section 2 with the difference that the communication bus is FlexRay.



1. Model the platform (two resources for processors and two tasks A and B) without FlexRay.
2. Initialising FlexRay bus

   i. new project → Add Bus Peripheral

   ii. In the 'Add Bus Peripheral Component' window, choose 'FlexRay Bus' as the peripheral name. Give a name, FlexRay_Bus.

   Now our platform has a FlexRay bus.

   iii. Choose the 'FlexRay_bus' under 'Buses' and you will see a 'Properties' tab with a lot of options. You could hover through them to see its description. Further details could be obtained from "vector_flexray.pdf" file shared with you.

Now we need to connect our Processors to the FlexRay bus. For this we need to add a FlexRay bus peripheral to each Processor.

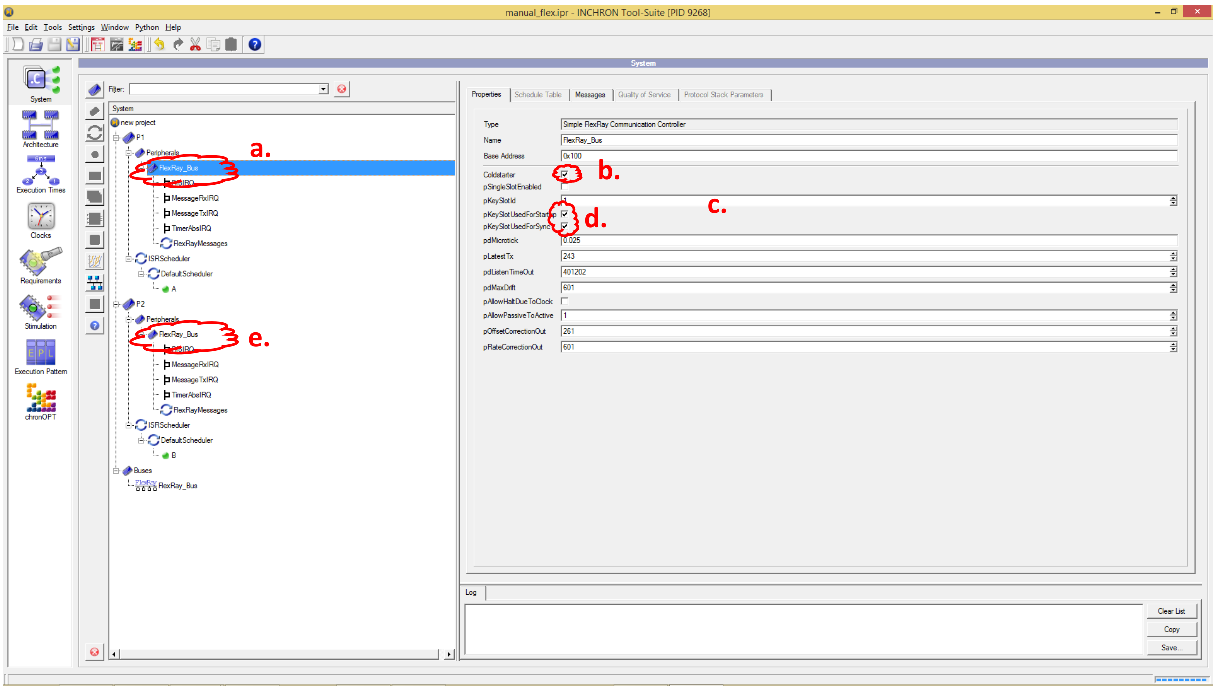
iv. Resource (P1 or P2) → Add Peripheral

v. In the 'Add Peripheral Component' window, choose 'Simple FlexRay Communication Controller' as 'peripheral type', a name (FlexRay_Bus) and a base address (0x100).

The above steps (iii – iv) need to be done for all processors connected to the FlexRay Bus. It is preferred to have different base addresses.

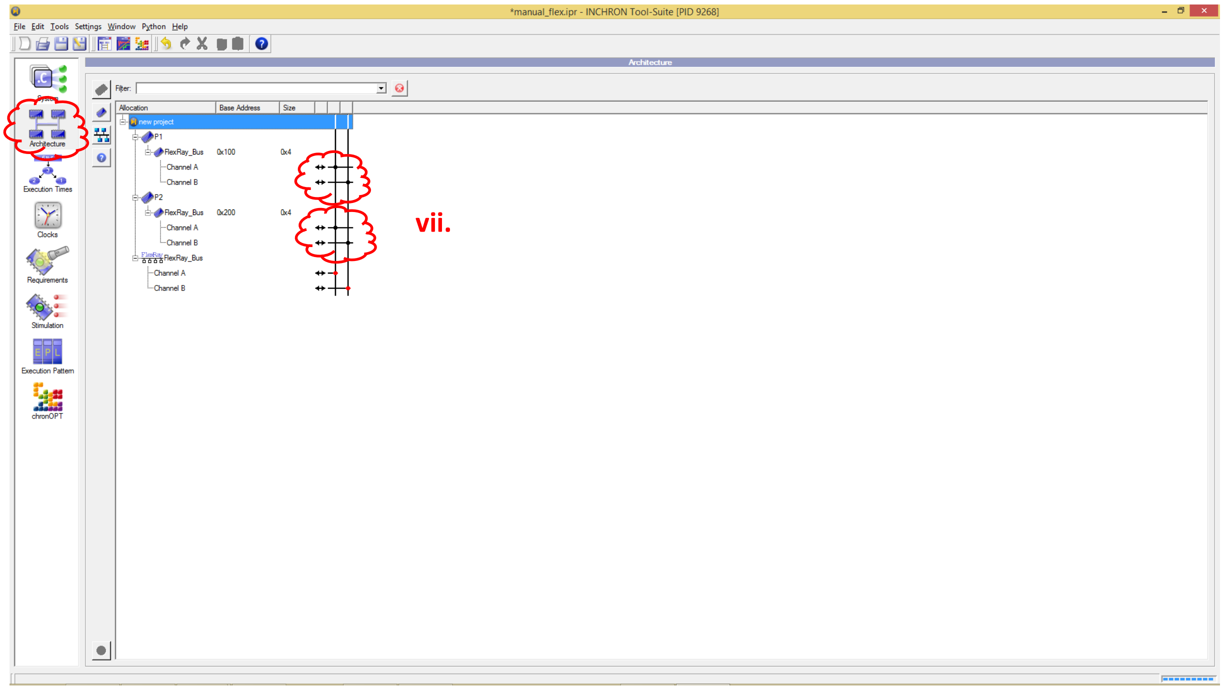
vi. FlexRay peripherals need to be configured. Choose FlexRay_Bus peripheral in P1. You would see a lot of options in its 'Properties' tab. There are few settings you need to do here. While using a FlexRay bus, one of the peripheral controller should act as a 'Coldstarter'. There should be one static slot ("pKeySlot") which should be used to transmit the 'startup' or 'sync' message and another static slot to receive it.

a. Choose FlexRay_Bus peripheral in P1
b. Check the 'Coldstarter' option
c. Assign a value to 'pKeySlotId'. The chosen ID should be less than the number of static slots ("gNumberOfStaticSlots") defined under 'properties' in 3.2.iii
d. Check 'pKeySlotUsedForStartup' and 'pKeySlotUsedForSync' options
e. Choose FlexRay_Bus peripheral in P2
f. Assign a value to 'pKeySlotId'. The chosen ID should be different than the one assigned to FlexRay_Bus peripheral in P1 (we assign value = 1 for P1 and =2 for P2)
g. Check 'pKeySlotUsedForStartup' and 'pKeySlotUsedForSync' options



For now, use default settings for other options.

vii. Connect the 'Simple FlexRay Communication Controller' in P1 and P2 to the FlexRay_bus in the 'Architecture' view.
Click on the graphical nodes between Channel A in P1 and P2 with the vertical line connected to Channel A of FlexRay bus. Do the same for Channel B.



viii. Now we need to add the FlexRay bus messages. Go back to 'System' view.

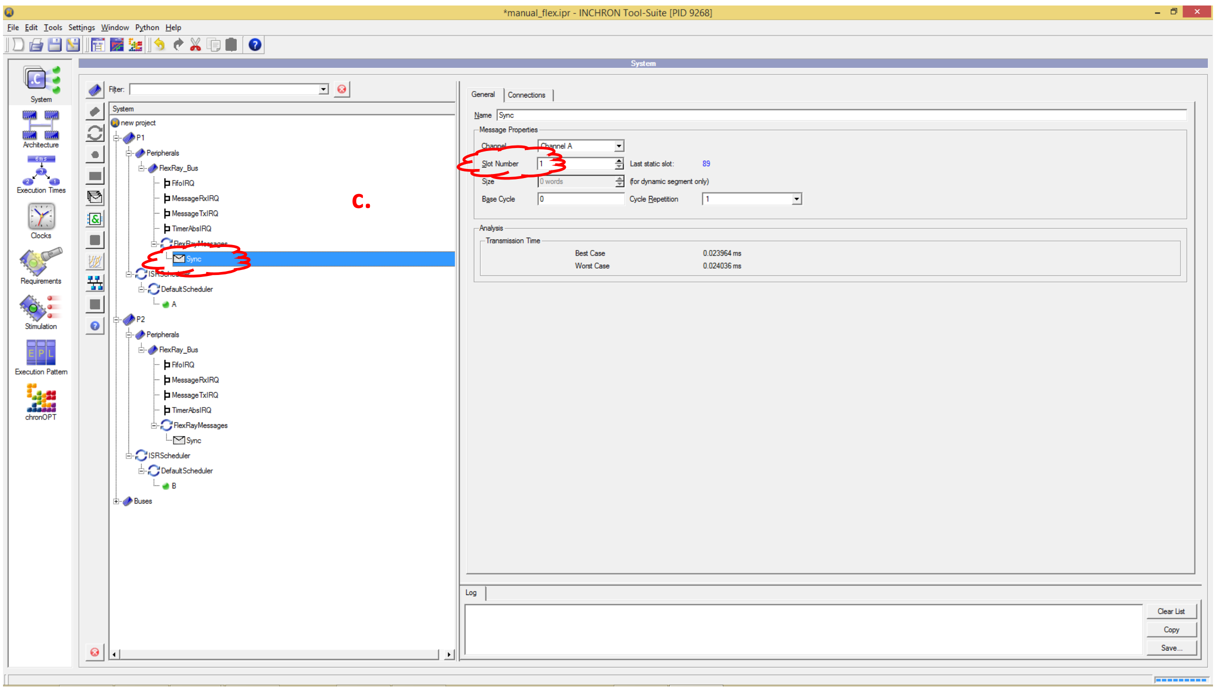
a. Choose P1

b. FlexRayMessages → Add Bus Message.

For the FlexRay bus to initialise, we need two startup and synchronisation bus messages (let's call it 'sync').

c. Choose 'sync' message. In the 'General' tab, choose 'Slot Number' of 'sync' message to be same as 'pKeySlotId' (=1 for P1 and =2 for P2, see 3.2.vi.f). Choose 'Channel A and B' for 'Channel' property. Use default options for other parameters.

d. Repeat steps b & c for P2.

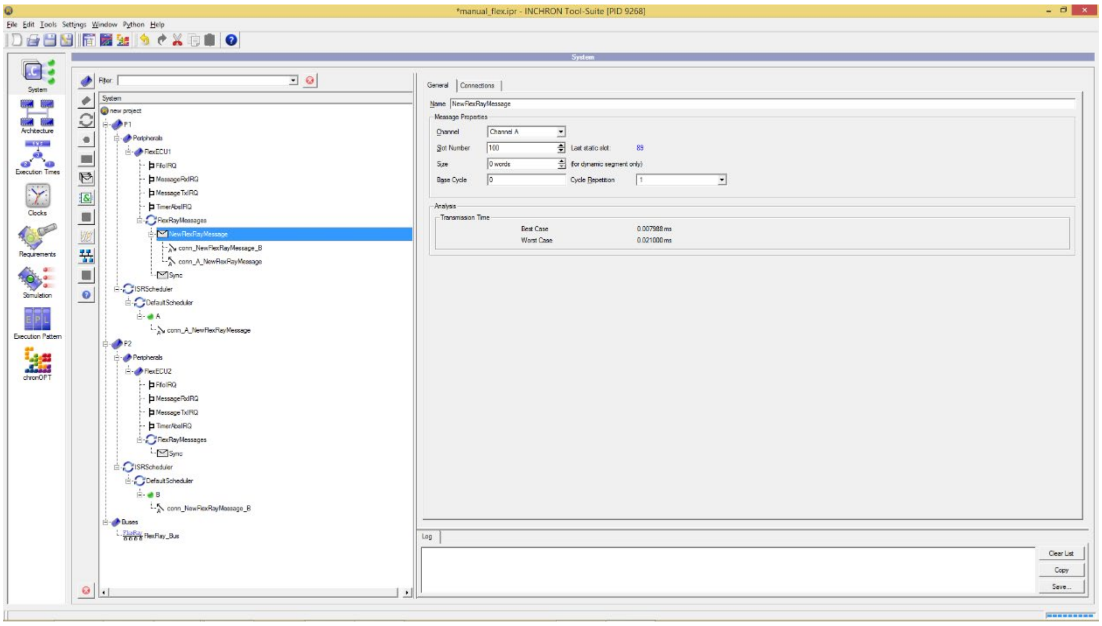


e. After adding ‘sync’ messages, add FlexRay message for sending message from A to B. Add a new bus message (‘NewFlexRayMessage’) and assign a ‘Slot Number’ for dynamic slot (=100). Use default options for other parameters.

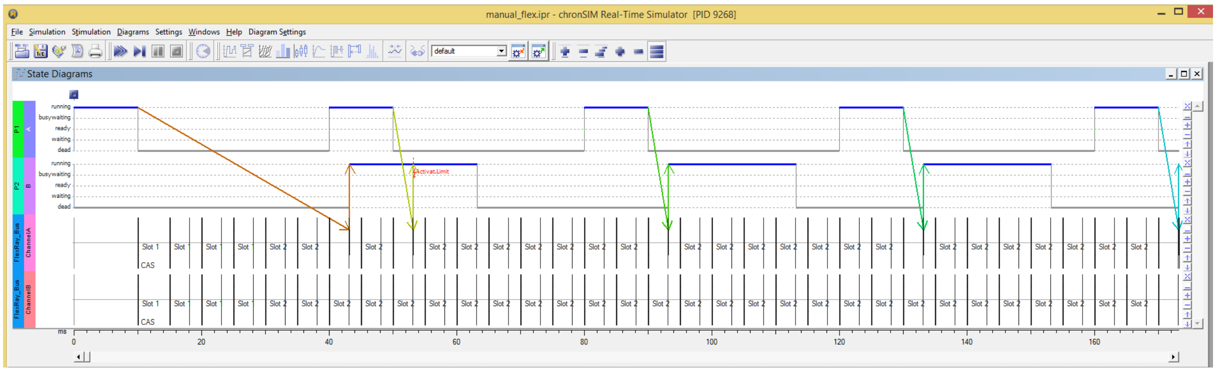
f. We can now send message from A to B through this ‘NewFlexRayMessage’. For this,

   i. Choose ‘Connections’ tab of ‘NewFlexRayMessage’.
   ii. Click on Process ‘A’ and
   iii. Choose ‘Add Incoming’.
   iv. Then, click on Process ‘B’ and
   v. Choose ‘Add Outgoing’.

NOTE: In FlexRay, there are a lot of parameters you need to consider. If your bus configuration is not correct, you may see unexpected simulation behaviour.

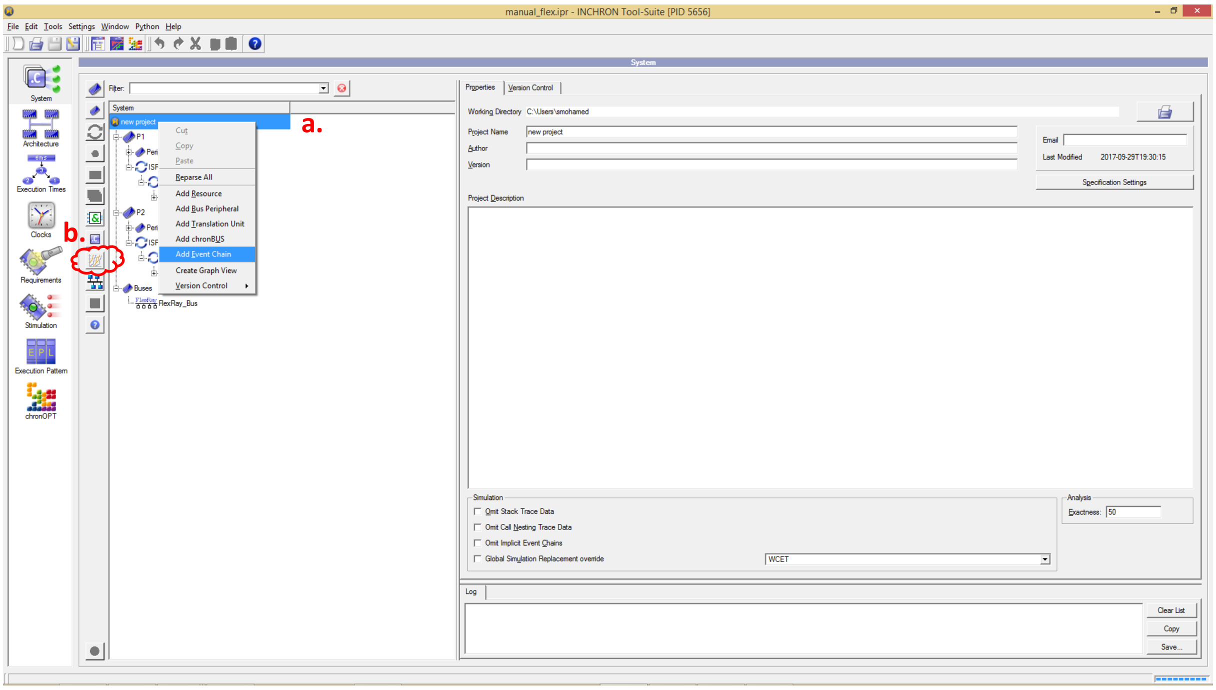


3. After doing all the above steps (in 3.2), you can simulate as explained in 1.7. (NOTE: you should have a stimulation scenario as explained in 1.6).

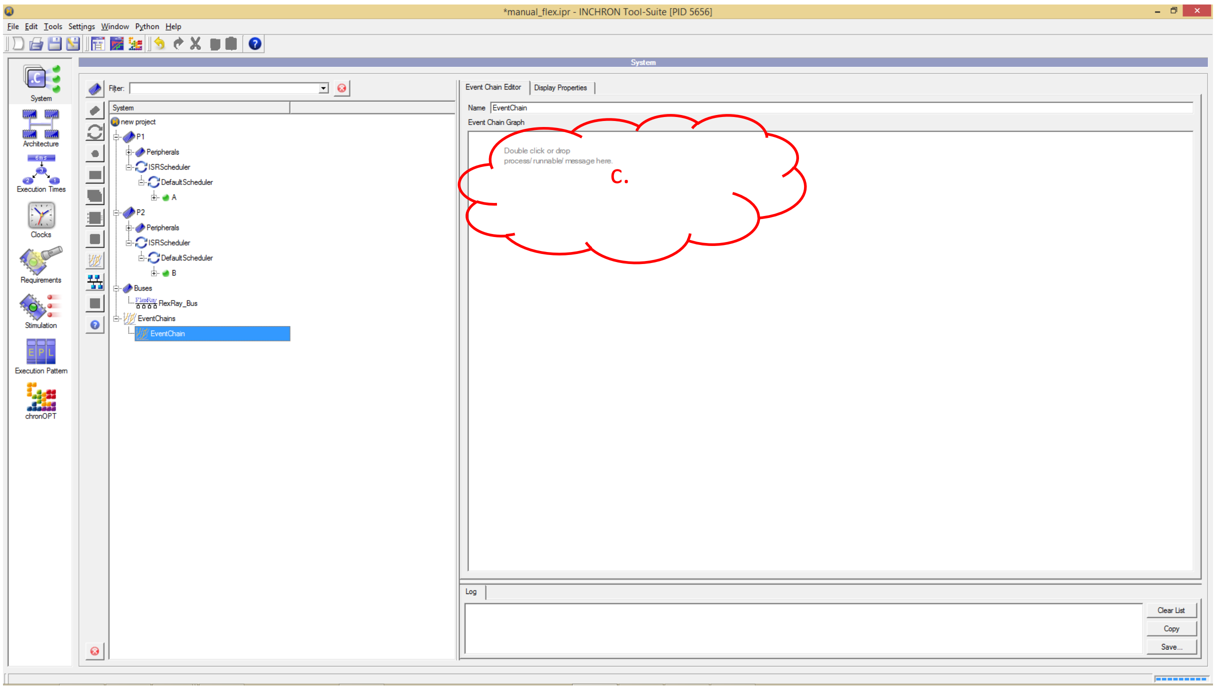


4. Adding Event Chain

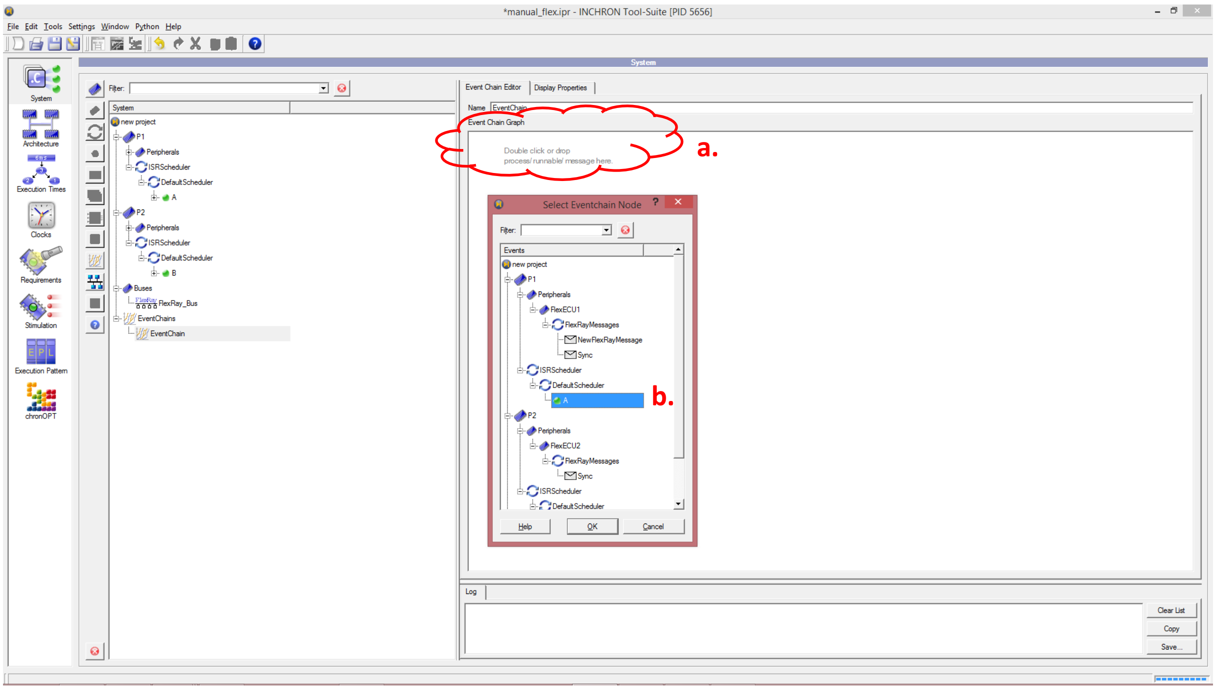
1. a. <Project_Name> → ‘Add Event Chain’ or alternatively click on icon b.



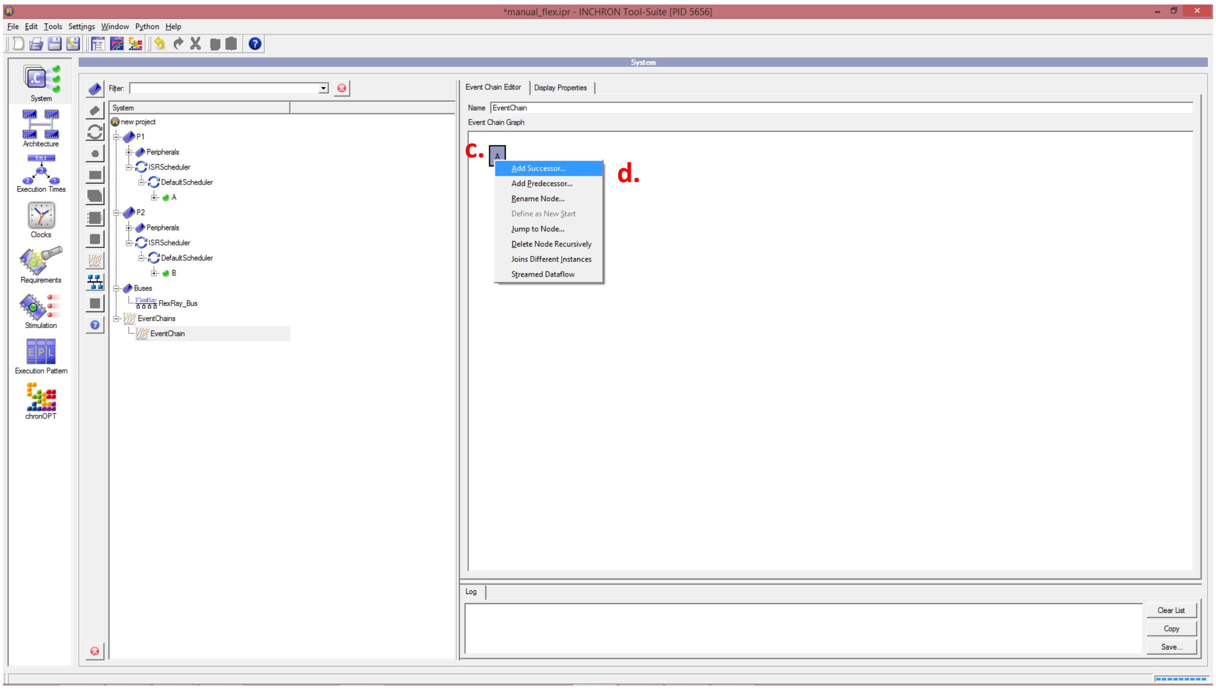
You will see the following window. ‘Drag and drop’ processes (or tasks) you want to consider in the event chain to c. alternatively, you could also double-click on the window to add process.



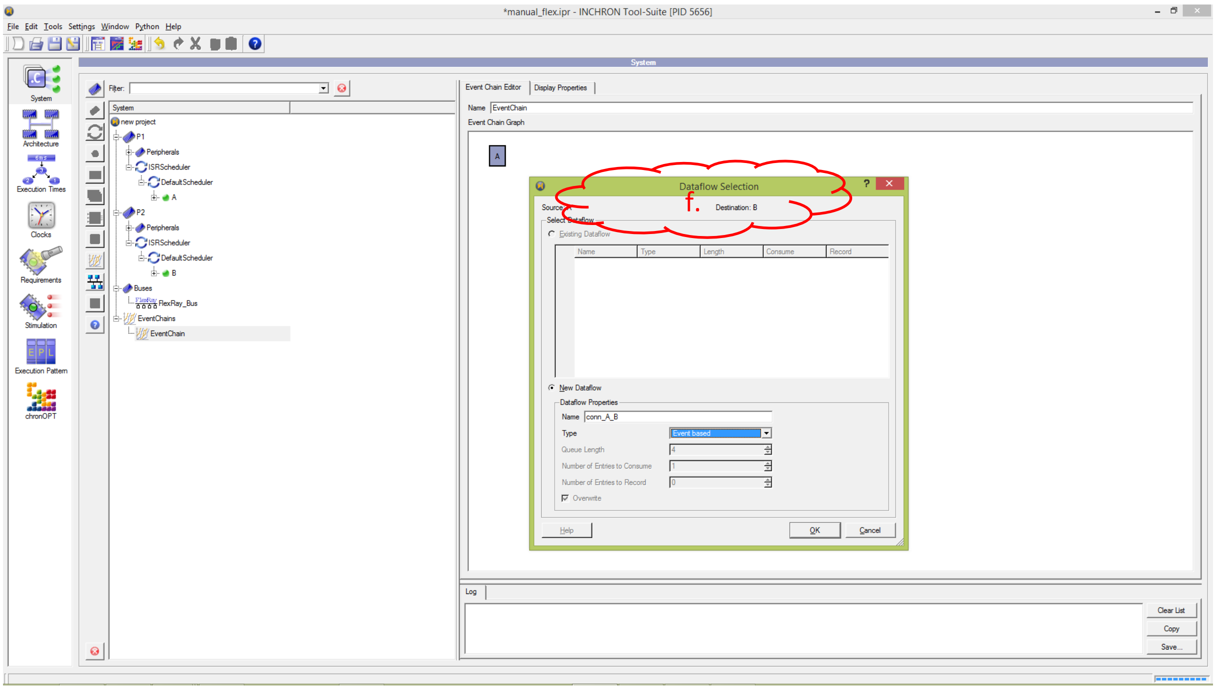
2. To consider an event chain from A to B, we need to do the following:
    a. Double-click on the ‘Event Chain Graph’ window



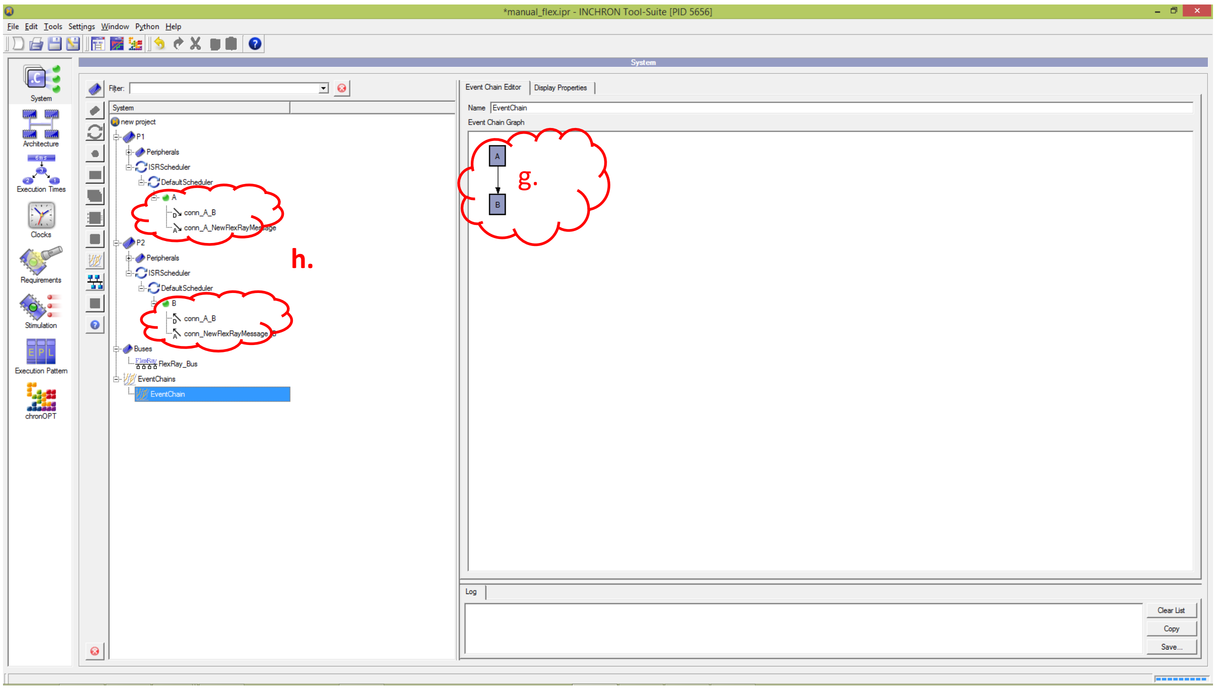
b. Choose process ‘A’ in the ‘Select Event Chain Node’ window and click OK.



c. Process A block in the ‘Event Chain Graph’ window.
d. Choose ‘Add Successor’ to add a successor node to A. You may use other options depending on your intent.
e. Choose process ‘B’ in the ‘Select Event Chain Node’ window and click OK.



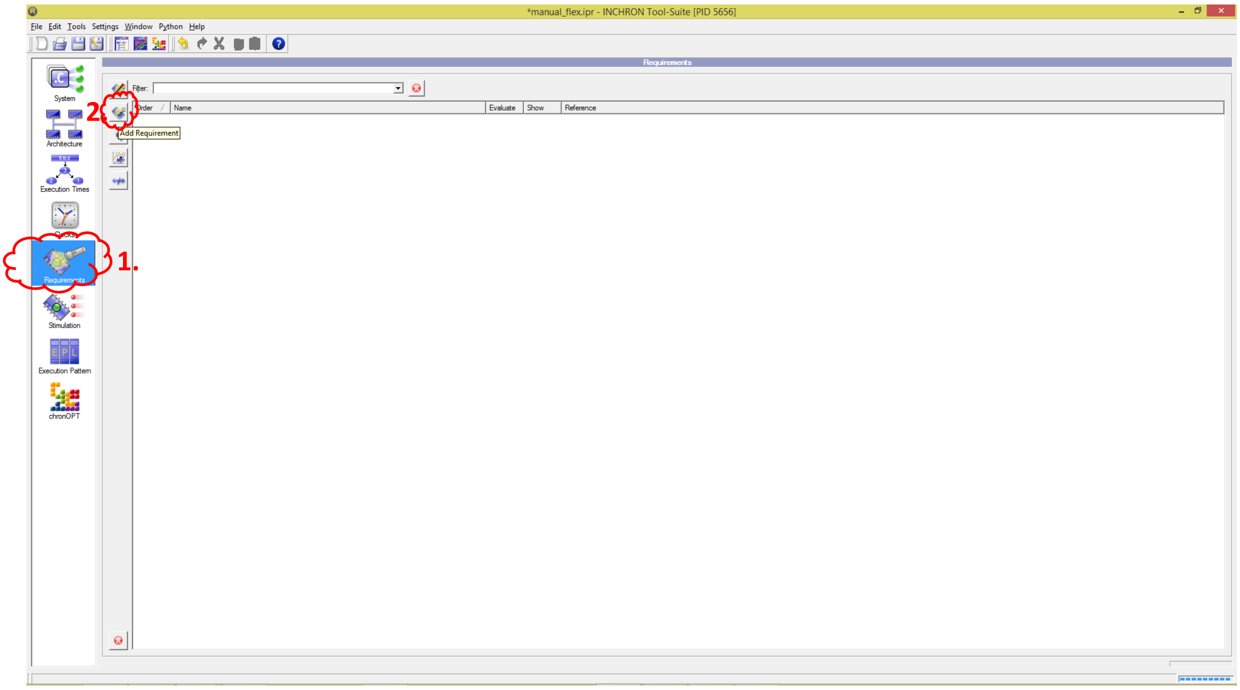
f. In the dataflow selection window that you see, use default options for now and click OK.
g. You would see the following event chain created with additional connections (h.) to represent the event chain in chronSIM and chronVAL.

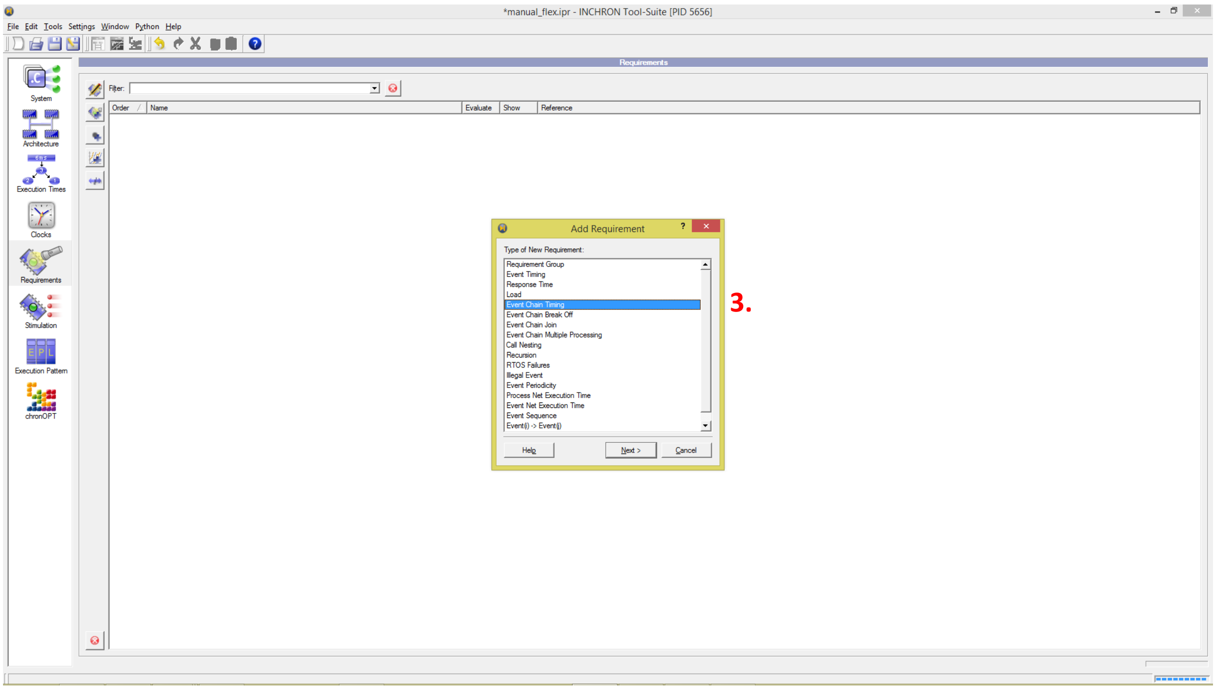


You can now visualise the event chains in chronSIM and chronVAL and also define requirements for this event chain.

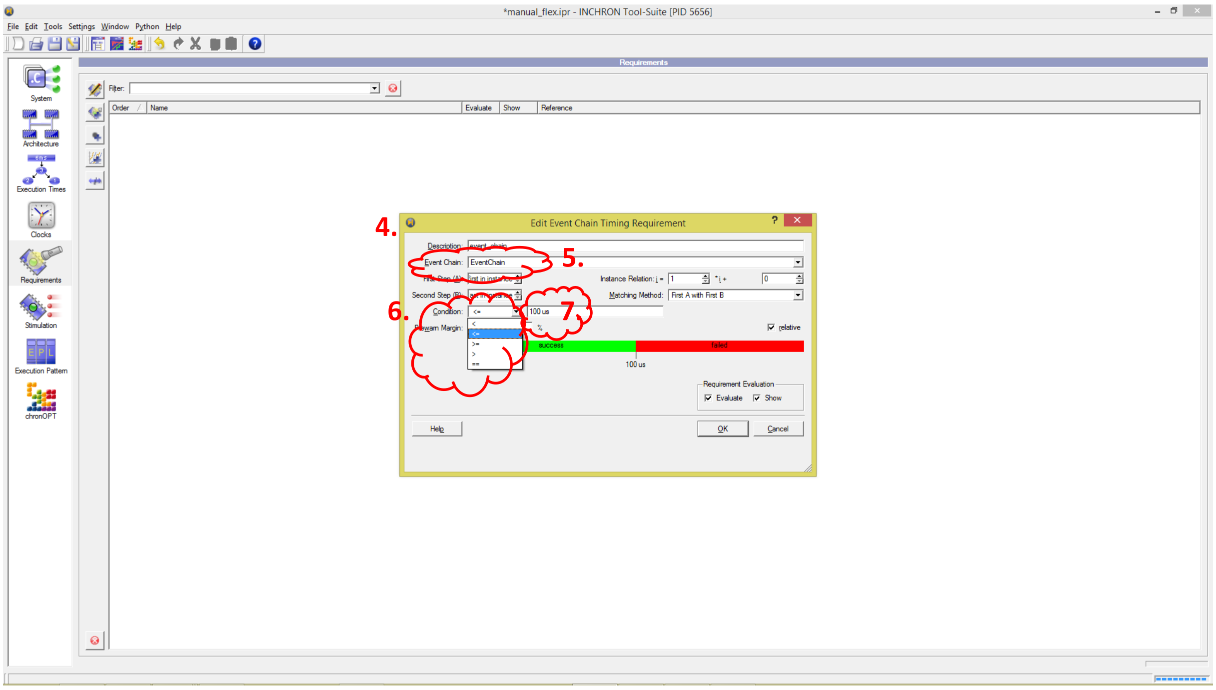
5. Adding Requirements

1. Choose requirements tab
2. Choose ‘Add Requirement’
3. Choose the relevant requirement you would like to evaluate in the ‘Add Requirement’ window. For this tutorial, let us choose ‘Event Chain Timing’ and click ‘Next’.

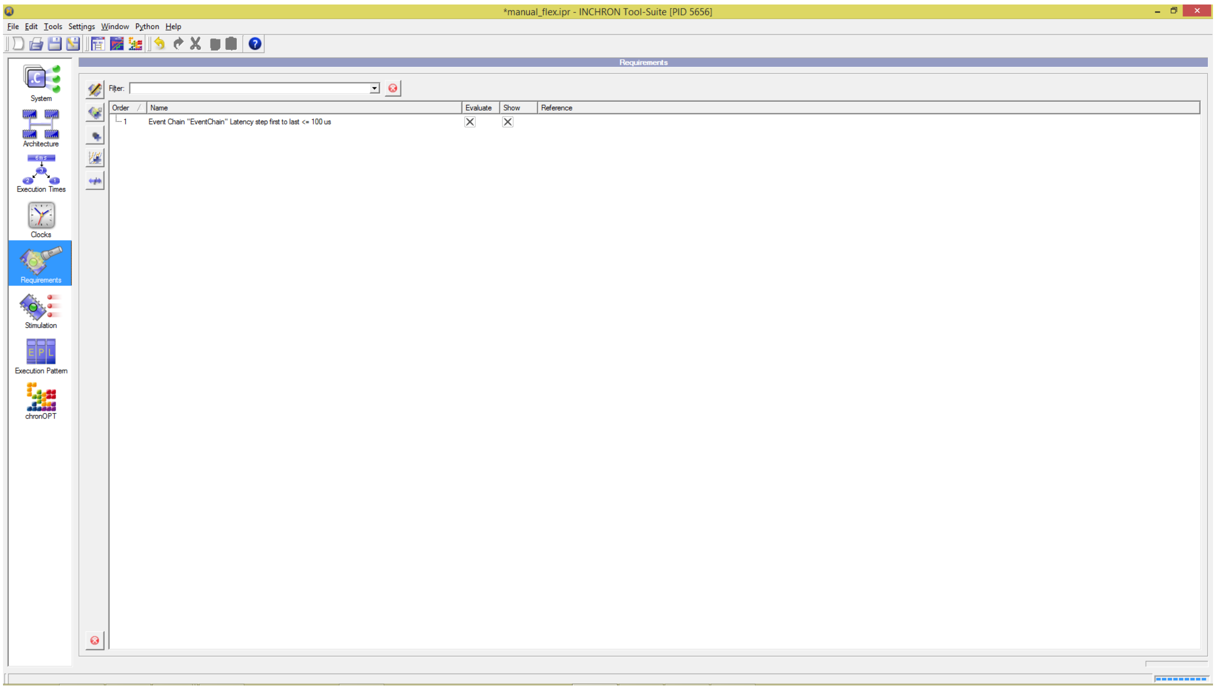




4. In the ‘Edit Event Chain Timing Requirement’ window, you could choose the options that you would need for your analysis. For this case, I would like to evaluate if the event chain meets its deadline.

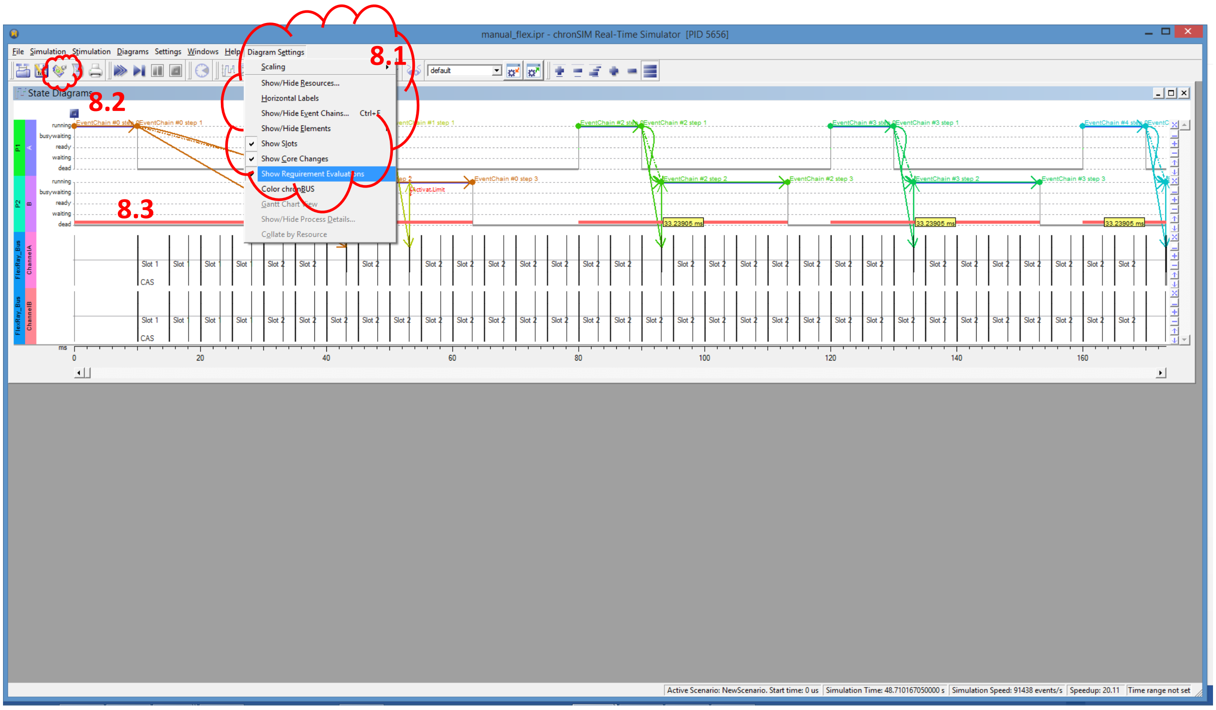


5. For this, first I choose the event chain to consider.
6. Then edit the condition to ‘<=’
7. Add the deadline timing and click OK.



The requirement is now added to the analysis.

8. You can now evaluate this requirement in the chronSIM simulation.
    1. To visualise go to Diagram Settings → Check ‘Show Requirement Evaluations’
    2. To modify and view the failed or successful (%) requirements, go to Windows → Edit Requirements. Alternatively, choose the icon.



3. Successful evaluations would be shown in green line and failed evaluations would be marked in red lines.