DECEMBER 16, 2020

LAB MANUAL – REAL TIME SYTSEMS

EXPERIMENT 1: INTRODUCTION TO CHEDDAR

BITS,PILANI – KK BIRLA GOA CAMPUS

Table of Contents

Introduction to Cheddar 2

Starting the Simulator: 2

Building a new project: 2

Running the Simulation: 5

Experiment 1: An Example 6

Lab Assignment 1 9

Q1 9

Q2 10

Q3 10

Q4 10

Q5 11

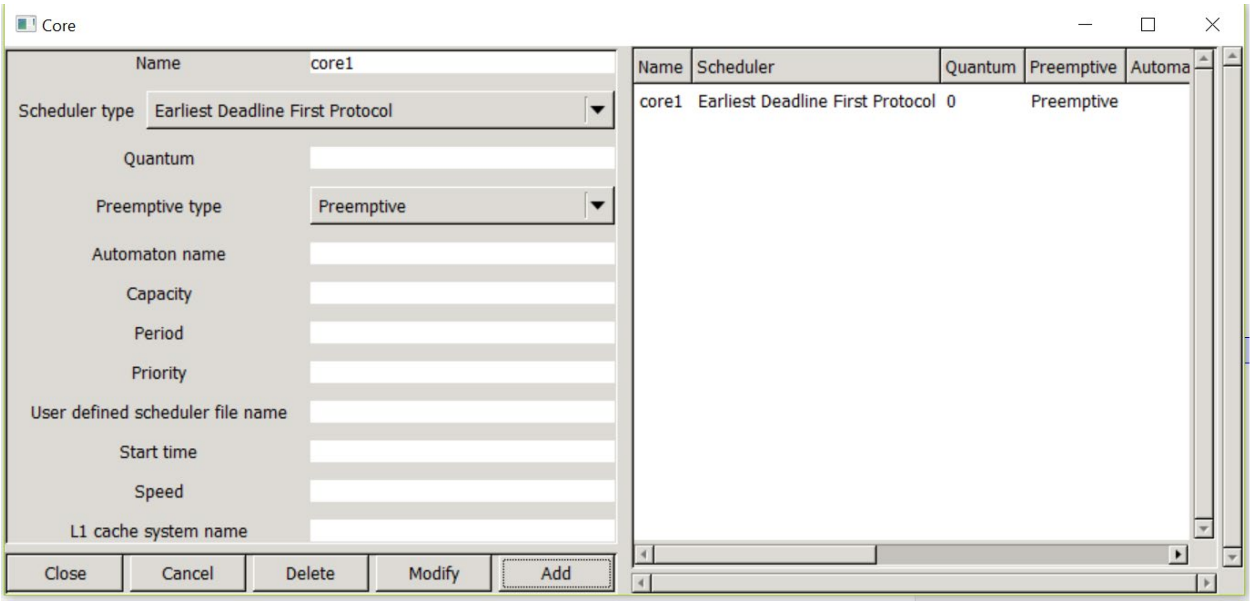
Introduction to Cheddar

Starting the Simulator:

Double click on the cheddar.exe from the ‘Cheddar-3.0-win32-bin’ folder. This will launch the application.

Building a new project:

1. From the file icon in the menu bar, choose ‘new’ XML project. All newly created files have to be saved in the current path with an extension of .sc.
2. **Adding a core:**
    - From the menu bar, Edit -> hardware -> Core.



Name core1

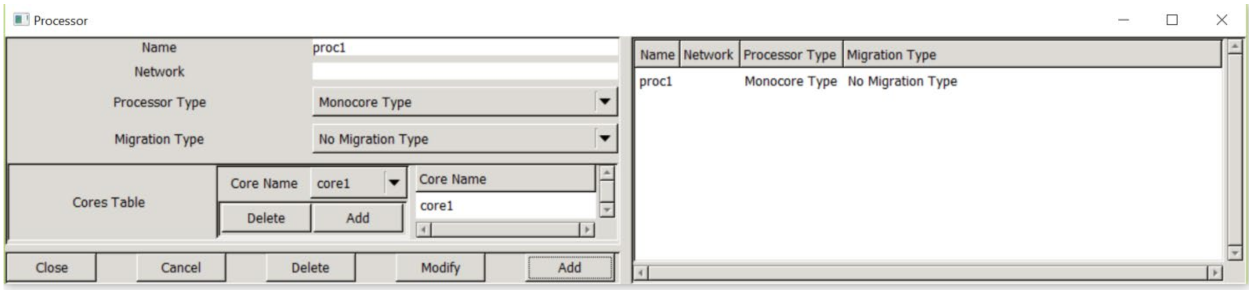
Scheduler type Earliest Deadline First Protocol

Preemptive type Preemptive

Name	Scheduler	Quantum	Preemptive	Automa
core1	Earliest Deadline First Protocol	0	Preemptive	

- Add the name of the core. For example, Core1, as shown in the figure.
- Scheduler type: Add the type of scheduling algorithm
- Preemptive type: Has 2 options:-preemptive and non-preemptive. To be chosen according to the requirement.
- Add the core by selecting the add option. The core name along with the parameters will be reflected on the right hand side.
- Modify the core parameters by selecting the core name and changing the parameter.
- To delete the core by choosing delete.

3. **Adding the Processor:**
   - From the menu bar, Edit -> Hardware -> Processor.



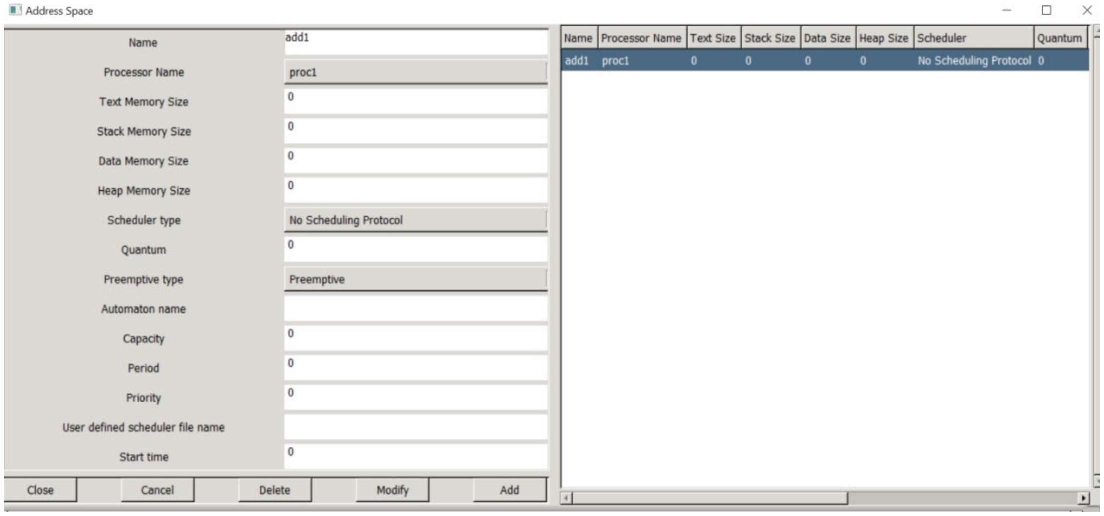
Name	proc1
Network	
Processor Type	Monocore Type
Migration Type	No Migration Type
Cores Table	Core Name: core1

Core Name
core1

Name	Network	Processor Type	Migration Type
proc1		Monocore Type	No Migration Type

   - Add the name of the processor.
   - Processor type:
     - Mono-core processor- Single core processor system.
     - Identical multicores- Multicore system with identical core parameters, that is, same architecture and same speed.
     - Uniform multicores- Same Architecture but different speeds.
     - Unrelated multicores- Different characteristics.
   - Migration Type:
     - No migration: The tasks will not be migrated from one processor to another.
     - Job level migration: The various instances of the task can be migrated across various processors.
     - Time unit migration: All instances of the task will be executed on the same processor.
   - Add the core from the 'cores table'.
   - The Processor name along with the other parameters will be reflected on the right hand side.

4. **Adding the address space:**
   - From the menu bar, Edit -> software -> Address space.

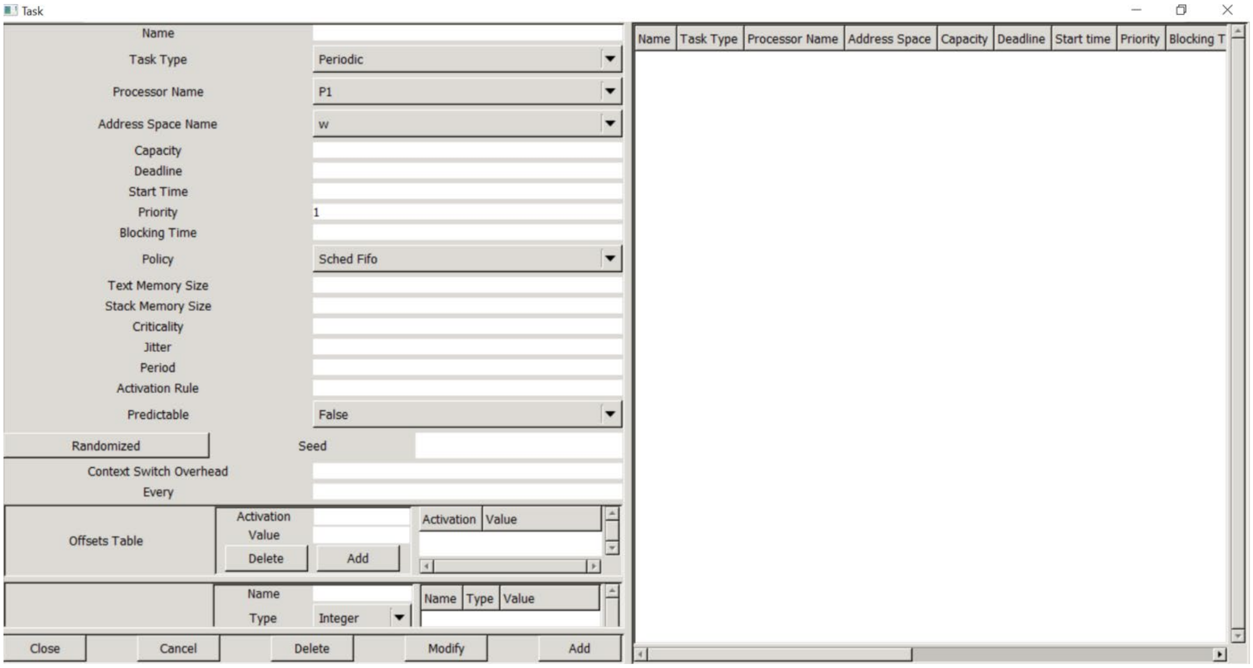


Name	add1
Processor Name	proc1
Text Memory Size	0
Stack Memory Size	0
Data Memory Size	0
Heap Memory Size	0
Scheduler type	No Scheduling Protocol
Quantum	0
Preemptive type	Preemptive
Automaton name	
Capacity	0
Period	0
Priority	0
User defined scheduler file name	
Start time	0

Name	Processor Name	Text Size	Stack Size	Data Size	Heap Size	Scheduler	Quantum
add1	proc1	0	0	0	0	No Scheduling Protocol	0

- ➢ Select the name for the address space.
- ➢ Processor name: Choose the processor. All the defined processors will appear in the drop down list.
- ➢ Scheduler type: An address space is a piece of memory which contains the processor parameters such as tasks and resources. A scheduler type can be defined for the address spaces in case of hierarchical scheduling algorithms, otherwise choose No Scheduling Protocol.
- ➢ Preemptive type: opt for non-preemptive scheduling in the address space.
- ➢ Add the address space.

5. **Adding a task:**
   - ➢ From the menu bar, Edit -> software -> task.



Task

Name	
Task Type	Periodic
Processor Name	P1
Address Space Name	w
Capacity	
Deadline	
Start Time	
Priority	1
Blocking Time	
Policy	Sched Fifo
Text Memory Size	
Stack Memory Size	
Criticality	
Jitter	
Period	
Activation Rule	
Predictable	False
Seed	
Context Switch Overhead	
Every	

Offsets Table: Activation, Value

Activation	Value

Name, Type: Integer

Name	Type	Value

Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Priority	Blocking T

- ➢ Add the name of the task.
- ➢ Task type:
  - Periodic task: The task will occur periodically after the time unit defined in the period field.
  - Aperiodic task: Tasks that will occur only once with soft deadline.
  - Sporadic task: Tasks that will occur only once with a hard deadline.
- ➢ Processor name: Drop down list contains all the processors defined. Choose the appropriate processor for the task.
- ➢ Capacity: Worst case Execution time of the task.
- ➢ Deadline: Absolute deadline. Refers to the completion time for the task.
- ➢ Start time: The time offset for the task, that is, the time at which the task will be ready for execution.

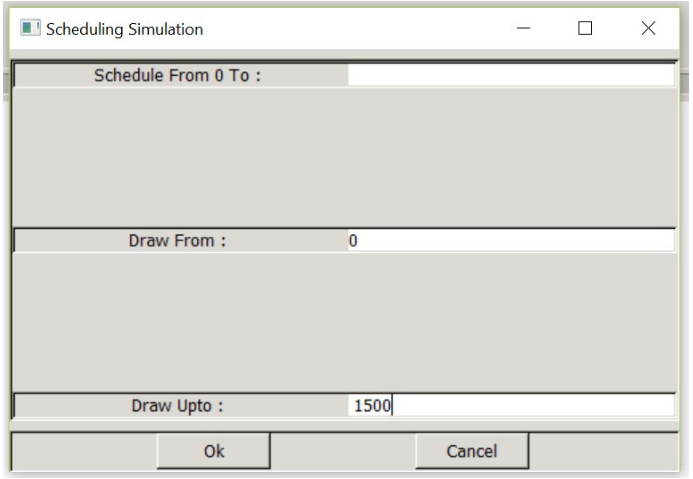
Priority: To define the priority of the task. For user defined priority, use POSIX scheduling (Highest priority first scheduling, Fixed priority). There are a total of 255 priority levels available. By default, 1- lowest priority and 255- highest priority. When 2 or more tasks have the same priority, the scheduler will execute the task depending upon the scheduling policy.

In case of other scheduling algorithms, priority is determined by the scheduler depending upon the task parameters such period, deadline, etc.

- Period: The periodicity of the task, that is, the time after which the task will reoccur. (note: Deadline must be lesser than the periodicity). Do not specify periodicity for aperiodic and sporadic task.

Running the Simulation:

- To run the simulation, click on ▷
- The following dialog box will appear, schedule from 0 to hyper-period.



Scheduling Simulation

Schedule From 0 To :

Draw From : 0

Draw Upto : 1500

- This will create a .xml file (eg: file1.xml) which will be present in the same path as .sc file.
- To clear the workspace, click on 🧹
- To set different colors to the task, From the menu bar choose, Tools->Scheduling->Scheduling options -> Display scheduling->Several colors for time tines.
- To check the schedulability feasibility tests, click on ⚙

Experiment 1: An Example

Consider the following task set. Scheduling algorithm: EDF

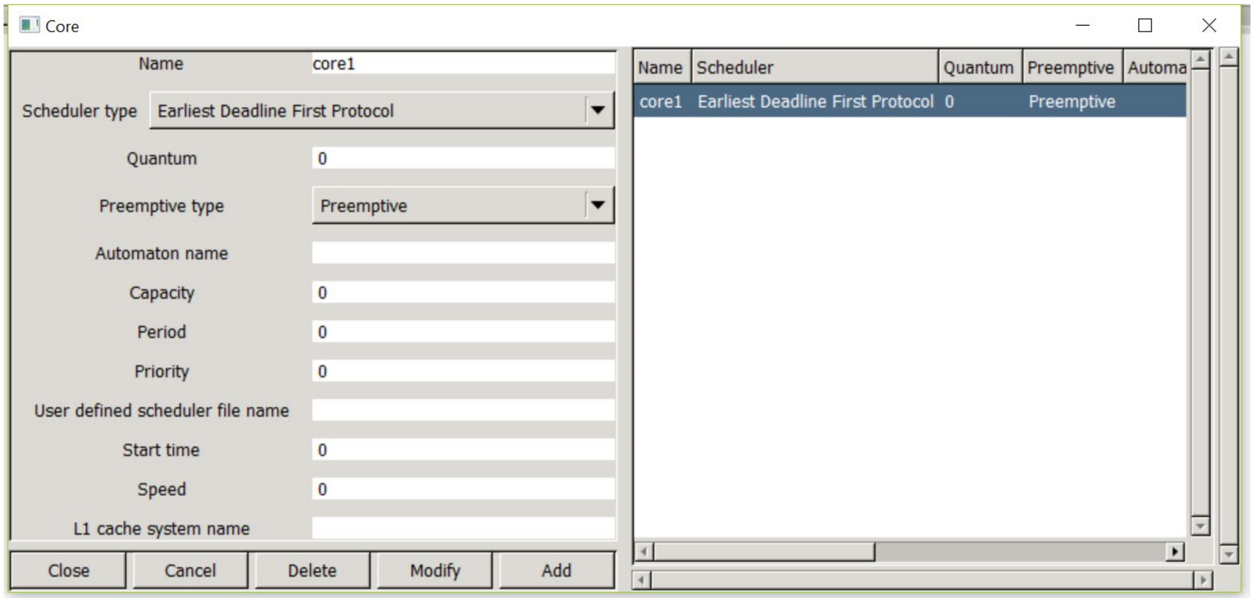
T1 = (1,4)

T2 = (2,5)

T3 = (1,20)

T4 = (2,20)

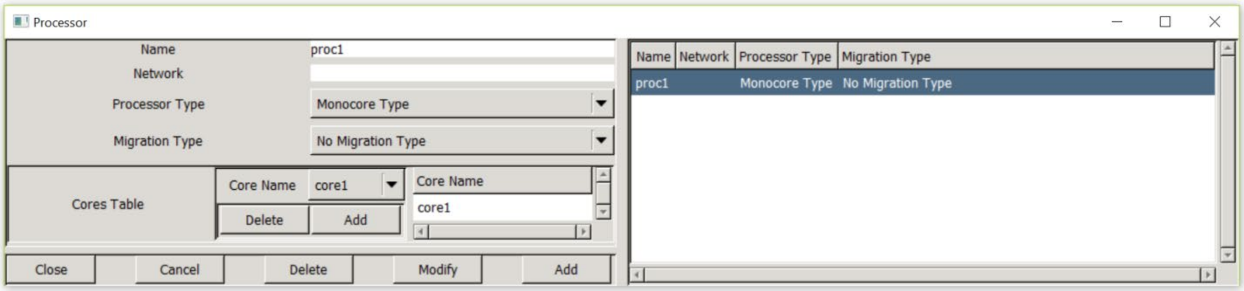
1. Adding the core:



Name	core1
Scheduler type	Earliest Deadline First Protocol
Quantum	0
Preemptive type	Preemptive
Automaton name	
Capacity	0
Period	0
Priority	0
User defined scheduler file name	
Start time	0
Speed	0
L1 cache system name	

Name	Scheduler	Quantum	Preemptive	Automa
core1	Earliest Deadline First Protocol	0	Preemptive	

2. Adding the processor

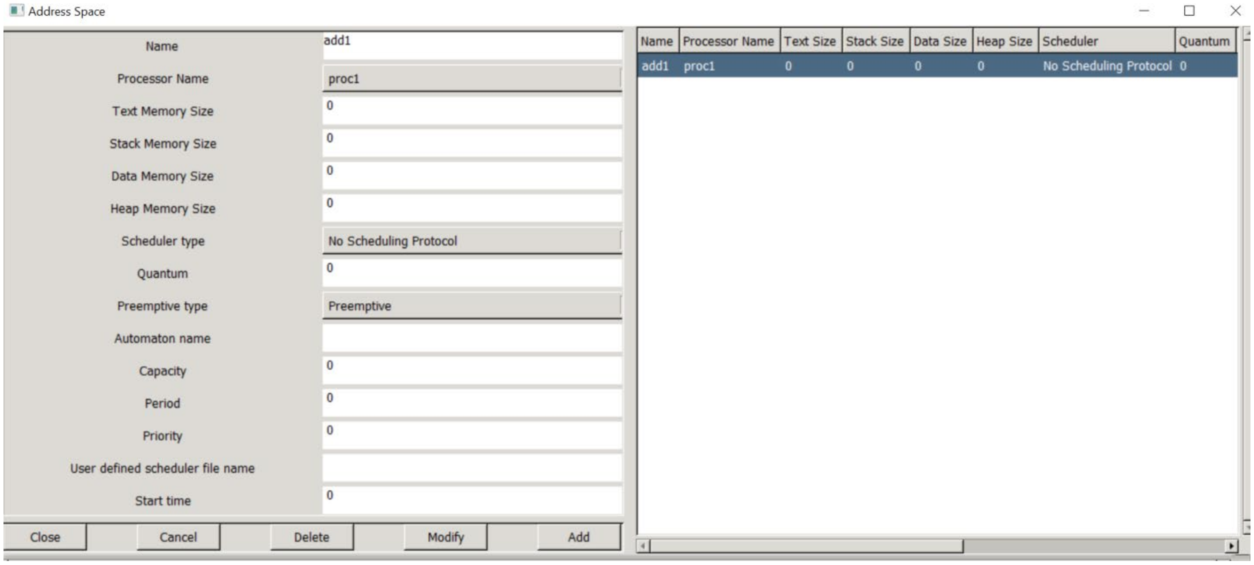


Name	proc1
Network	
Processor Type	Monocore Type
Migration Type	No Migration Type
Core Name	core1

Core Name
core1

Name	Network	Processor Type	Migration Type
proc1		Monocore Type	No Migration Type

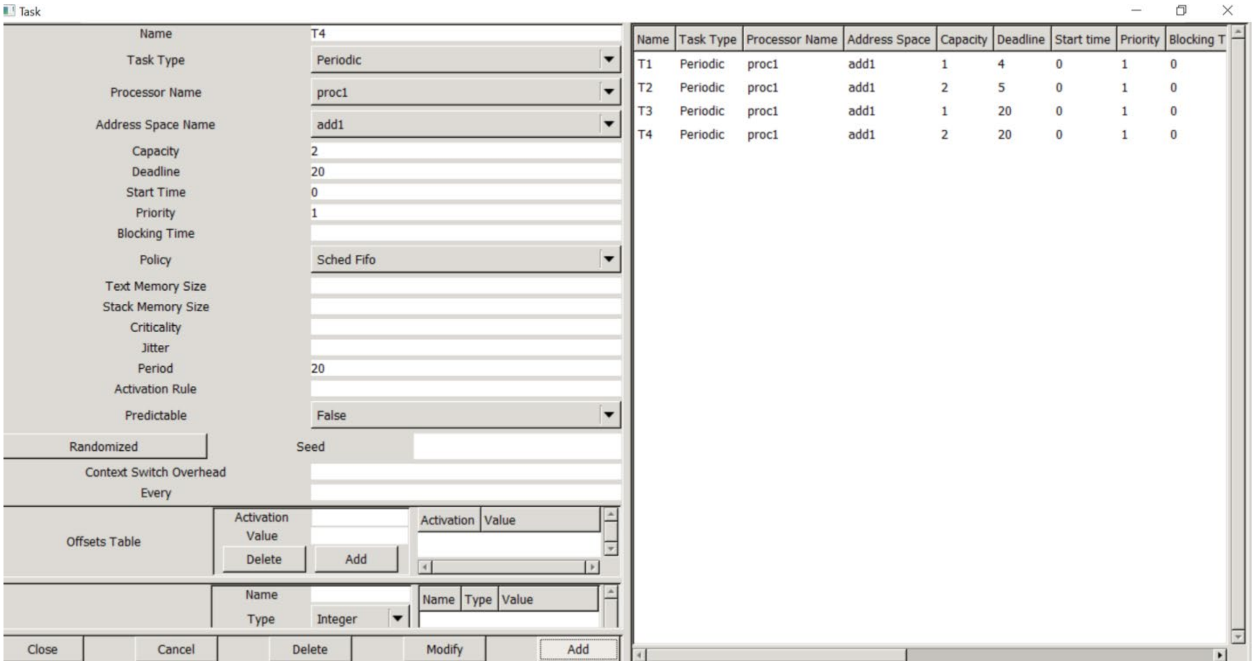
3. Adding the address space:



Name	add1
Processor Name	proc1
Text Memory Size	0
Stack Memory Size	0
Data Memory Size	0
Heap Memory Size	0
Scheduler type	No Scheduling Protocol
Quantum	0
Preemptive type	Preemptive
Automaton name	
Capacity	0
Period	0
Priority	0
User defined scheduler file name	
Start time	0

Name	Processor Name	Text Size	Stack Size	Data Size	Heap Size	Scheduler	Quantum
add1	proc1	0	0	0	0	No Scheduling Protocol	0

4. Adding the tasks:



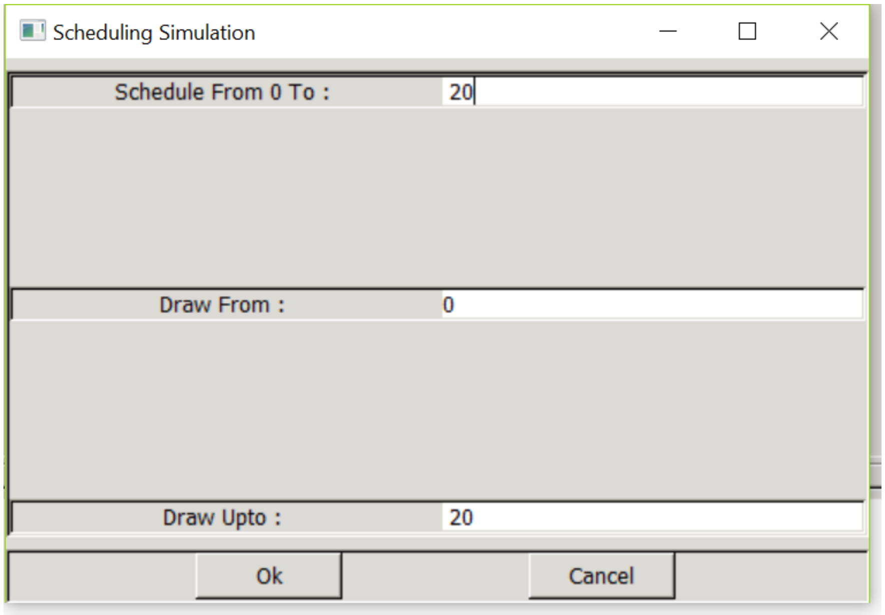
Name	T4
Task Type	Periodic
Processor Name	proc1
Address Space Name	add1
Capacity	2
Deadline	20
Start Time	0
Priority	1
Blocking Time	
Policy	Sched Fifo
Text Memory Size	
Stack Memory Size	
Criticality	
Jitter	
Period	20
Activation Rule	
Predictable	False
Seed	
Context Switch Overhead	
Every	
Activation	
Value	
Name	
Type	Integer

Activation	Value

Name	Type	Value

Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Priority	Blocking T
T1	Periodic	proc1	add1	1	4	0	1	0
T2	Periodic	proc1	add1	2	5	0	1	0
T3	Periodic	proc1	add1	1	20	0	1	0
T4	Periodic	proc1	add1	2	20	0	1	0

5. Run the simulation.

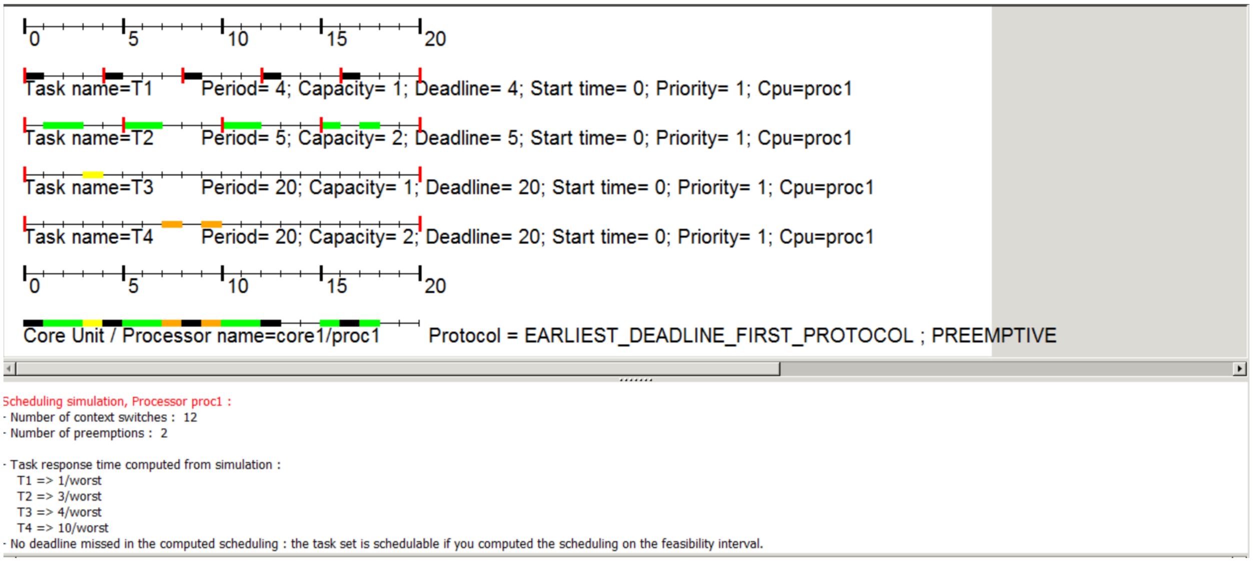


Schedule From 0 To : 20

Draw From : 0

Draw Upto : 20

Results obtained:



```
Task name=T1    Period= 4; Capacity= 1; Deadline= 4; Start time= 0; Priority= 1; Cpu=proc1
Task name=T2    Period= 5; Capacity= 2; Deadline= 5; Start time= 0; Priority= 1; Cpu=proc1
Task name=T3    Period= 20; Capacity= 1; Deadline= 20; Start time= 0; Priority= 1; Cpu=proc1
Task name=T4    Period= 20; Capacity= 2; Deadline= 20; Start time= 0; Priority= 1; Cpu=proc1
Core Unit / Processor name=core1/proc1    Protocol = EARLIEST_DEADLINE_FIRST_PROTOCOL ; PREEMPTIVE
```

```
Scheduling simulation, Processor proc1 :
- Number of context switches :  12
- Number of preemptions :  2

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 4/worst
   T4 => 10/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

According to EDF, the tasks are given the priorities as follows: T1 > T2 > T3 = T4. Though T3 and T4 have the same priority, T3 will be scheduled first as it follows a scheduled FIFO policy.

At time unit 7, T4 (deadline = 20) will start execution, but at t = 8, third instance of T1 will arrive which has a deadline of 12, hence it preempts T4. T4 will continue its execution at t = 9.

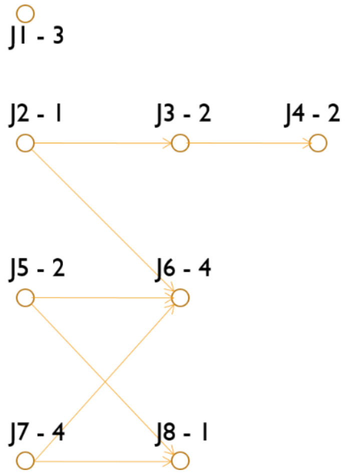
Similarly, at t = 16, T1 will preempt T2 as it has lower deadline (hence higher priority) in comparison to T2.

Lab Assignment 1

For the following Task sets and scheduling algorithms – submit the screen shoot of the schedule generated by Cheddar – with proper interpretations and conclusions regarding the schedule generated. If the task set is not schedulable – suggest an alternate method of scheduling

Q1

For the task graph show below show that a priority driven schedule with non-preemption will produce a better schedule than priority driven schedule with pre-emption



All tasks are aperiodic with a deadline of 12. All tasks except for J5 release at 0. J5 releases at 4. The priority of jobs is such that J1 has a higher priority when compared to J2 has a higher priority when compared to J3 and so on. The scheduling is done on a processor which has dual identical cores with jobs migratable at any point in time.

Q2

Show that the following tasks table is schedulable using EDF on a mono-core processor, if pre-emption is allowed but is not schedulable if pre-emption is not allowed. All tasks are aperiodic.

	J1	J2	J3
r	0	2	4
e	3	6	4
d	10	14	12

Q3

Show that the following tasks table is schedulable using LLF but not using EDF on a dual-core processor. All tasks are aperiodic.

	J1	J2	J3
r	0	0	0
e	1	1	5
d	1	2	5

Q4

Show that the following tasks table causes indeterminism due to the execution time – when scheduled using EDF. All tasks are periodic

	r	d	e
J1	0	10	5
J2	0	10	2-6
J3	4	15	8
J4	0	20	10

Q5

Show that the following tasks are not schedulable using either EDf/LLF on a tri-core processor. All tasks are periodic and preemptable. Migration is not possible between a Job.

	A	**B**	**C**	**D**	**E**
ϕ	0	0	0	0	0
e	1	1	1	6	6
p	2	2	2	8	8