CST 8703 Lab 3 - Timing Diagrams and Scheduling Feasibility

Kyle Chisholm

2022-05-01

Real-Time Systems and Embedded Programming

Feasibility and worst case scenarios for a simulated workload are analyzed with programs running on the pi and verified with Cheddar, a real-time scheduling analysis tool.

Source code available at https://github.com/chishok/CST8703-Lab3.

Submission Deadline: July 3, 2022

Background

When designing a real-time system, modeling and analysis tools can be used to determine the feasibility of meeting all required deadlines. For control systems, there are often multiple periodic tasks each with a deadline, worst-case execution time, and priority. The ability to execute every task before their deadlines expire depend on the strategy employed by the system to schedule when to provide cpu resources for each task (thread).

Feasibility analyses for periodic tasks have the following assumptions (as expressed by [1]):

- The instances of a periodic task τ_i are regularly activated at a constant rate. The interval T_i between two consecutive activations is the period of the task.
- All instances of a periodic task τ_i have the same worst-case execution time C_i.
- All instances of a periodic task τ_i have the same relative deadline D_i which is equal to the period T_i .
- All tasks in a system in a set of tasks Γ are independent; that is, there are no precedence relations and no resource constraints.
- No task can suspend itself, for example on I/O operations.
- All tasks are released as soon as they arrive.
- All overheads in the kernel are assumed to be zero.

In real-world applications, all the assumptions cannot be realistically satisfied and attempting to enforce a rigid timing schedule can be problematic. In practice, safety margins should be put in place when designing and modeling the real-time system. Additionally, a real-time feasibility analysis complements a broader and more rigorous testing strategy including unit testing, functional testing, benchmarking, profiling, integration testing, life testing, regression testing, etc.

A simple preemptive scheduling algorithm for periodic tasks is called **Rate Monotonic** which assigns priorities P_i to each task τ_i in a task set $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ such that tasks are ordered by increasing periods and decreasing priority. The first task τ_1 will have the highest priority and shortest period, and the last task τ_n has the lowest priority and longest period.

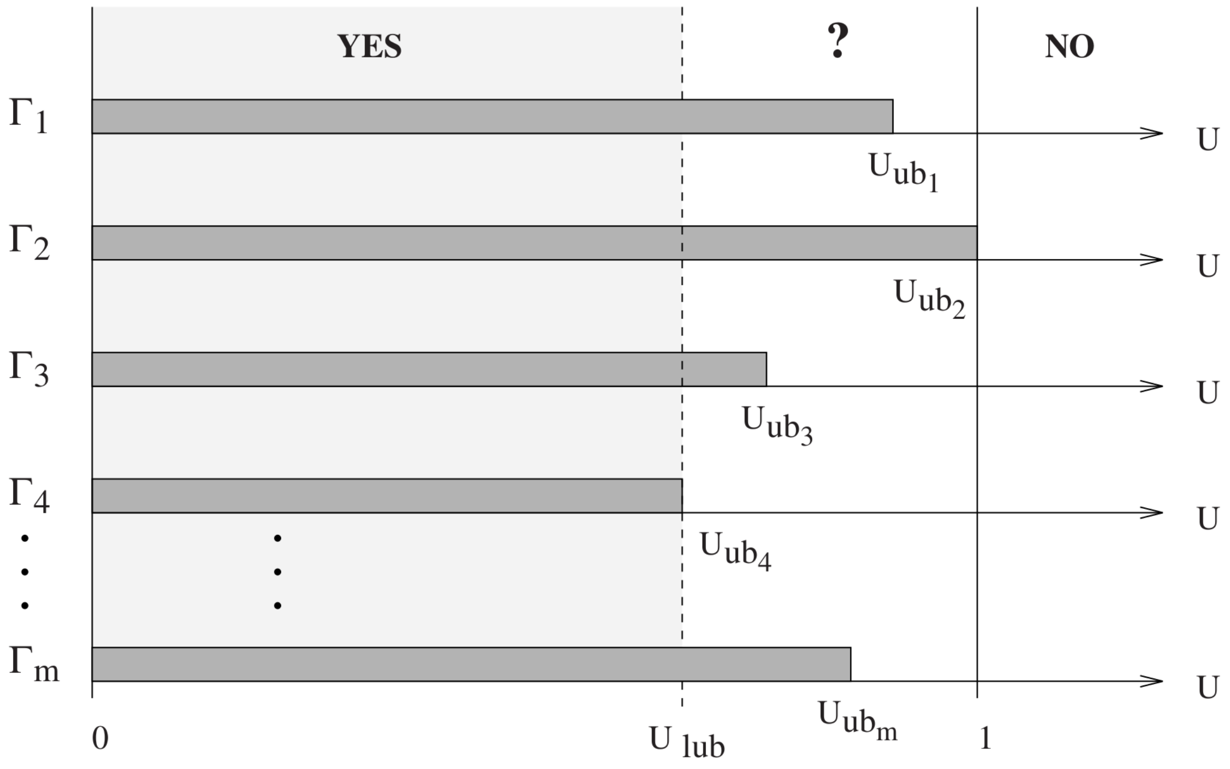
There are several methods of determining the feasibility of a set of periodic tasks with fixed period and worst-case execution times. A conservative approach for rate monotonic scheduling is to check whether the total utilization factor U is less than the "lower upper bound" (See [1], Chapter 4.3.3). The total utilization factor for n tasks is calculated as

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i},$$

which indicates that a factor greater than 1.0 is not feasible for any algorithm since it will consume more execution time than is available in all time periods. However, a value less than 1.0 may or may not be feasible. Any utilization factor U less than the least upper bound utilization factor U_{lub} is guaranteed to be feasible for a task set. The least upper bound utilization factor (derived in [1] Chapter 4.3.3) is

$$U_{lub} = n(2^{1/n} - 1).$$

For factors in the range $U_{lub} < U \leq 1.0$ feasibility is still uncertain. The following figure illustrates the concept of least upper bound and feasiblity of various task sets:

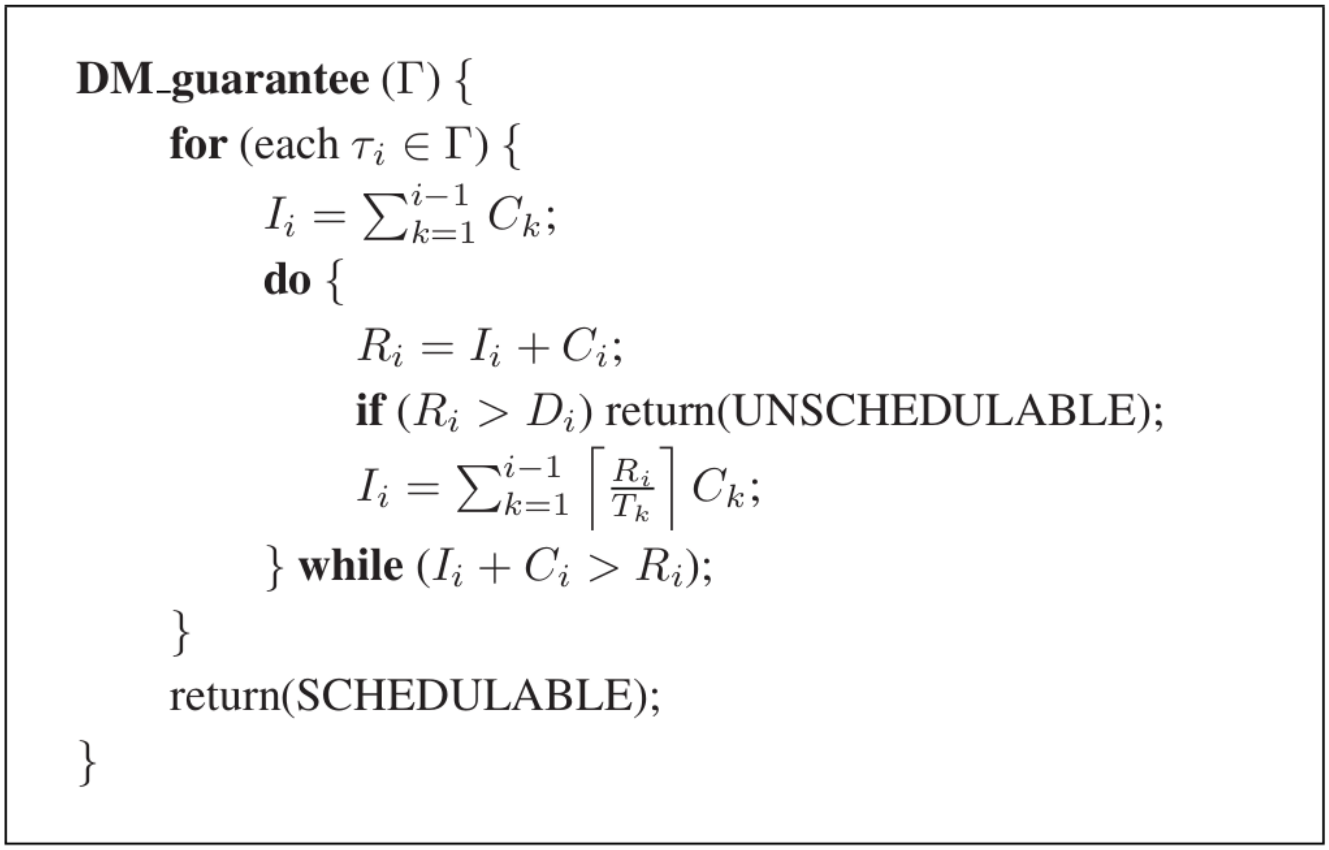


Relationship of feasibility to least upper bound of processor utilization factor.

A less conservative least upper bound for rate monotonic scheduling is the hyperbolic bound where the task set is feasible under the following condition:

$$\prod_{i=1}^{n}(U_i + 1) < 2.$$

If the least upper bound and hyperbolic bound functions are not sufficient to determine feasibility, a deadline monotonic schedule (See [1] Chapter 4.5) algorithm can be employed called "response time analysis" (Figure 1). Deadline monotonic is an extension of rate monotonic and they are equivalent if the deadline D is equal to the period T for each task i, such that $D_i = T_i$. Figure 1 provides the response time analysis algorithm. The "response time analysis" algorithm provides an exact solution to the feasibility problem but is less efficient computationally compared to the above metrics.



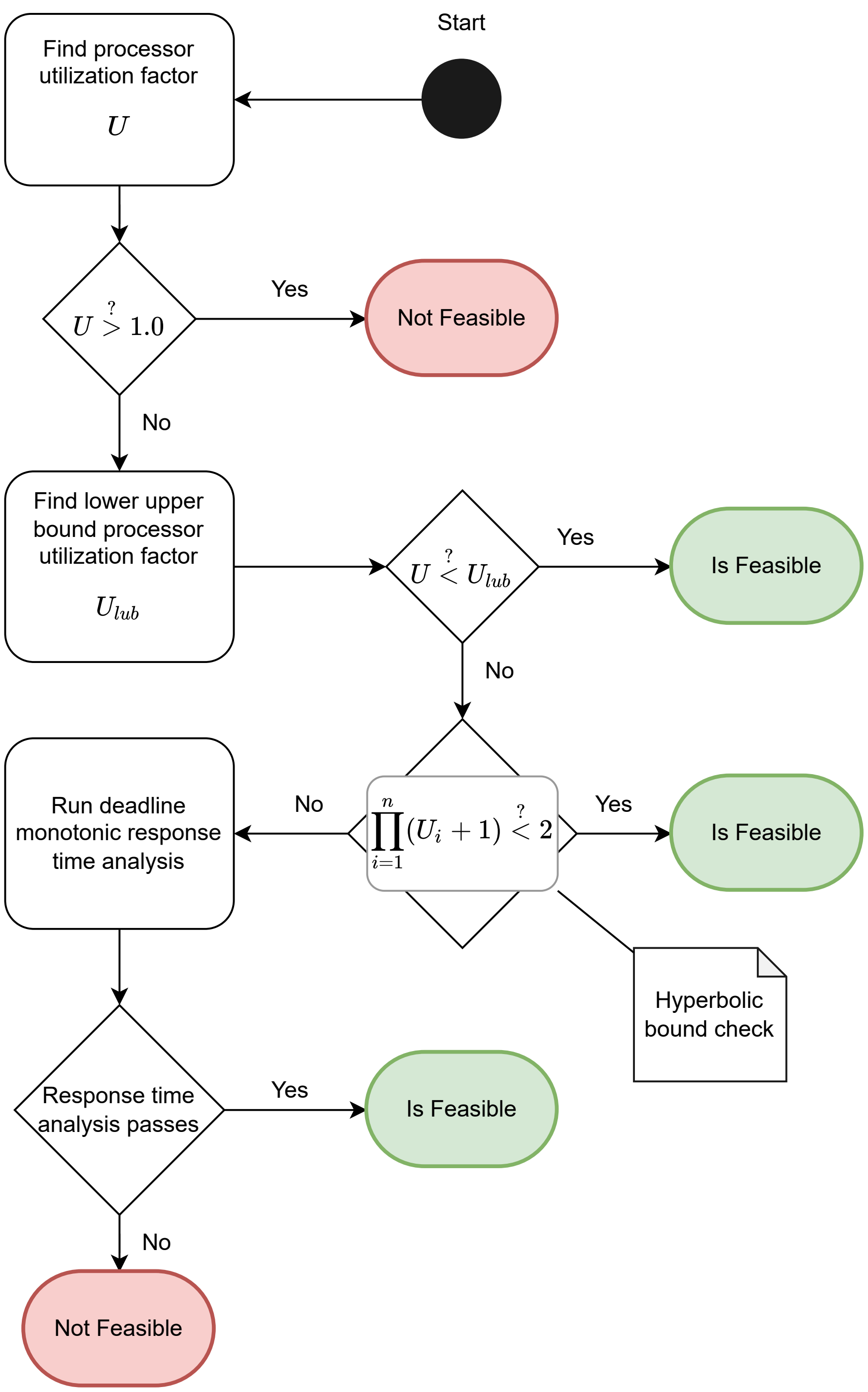
```
DM_guarantee ($\Gamma$) {
    for (each $\tau_i \in \Gamma$) {
        $I_i = \sum_{k=1}^{i-1} C_k$;
        do {
            $R_i = I_i + C_i$;
            if ($R_i > D_i$) return(UNSCHEDULABLE);
            $I_i = \sum_{k=1}^{i-1} \left\lceil \frac{R_i}{T_k} \right\rceil C_k$;
        } while ($I_i + C_i > R_i$);
    }
    return(SCHEDULABLE);
}
```

Figure 1: Response time analysis algorithm ([1] Figure 4.17).

Putting it all together, Figure 2 provides a flowchart for determining rate monotonic feasibility of a task set.

Materials

1. Raspberry Pi 4.
2. Desktop computer with Linux, Windows, or Mac operating system.



Start
Find processor utilization factor U
$U \overset{?}{>} 1.0$
Yes
Not Feasible
No
Find lower upper bound processor utilization factor U_{lub}
$U \overset{?}{<} U_{lub}$
Yes
Is Feasible
No
$\prod_{i=1}^{n}(U_i + 1) \overset{?}{<} 2$
Yes
Is Feasible
No
Hyperbolic bound check
Run deadline monotonic response time analysis
Response time analysis passes
Yes
Is Feasible
No
Not Feasible

Figure 2: Rate monotonic feasibility analysis flowchart.

3. Wired or wireless local network.

Methods

Prerequisites

Clone source from GitHub repository https://github.com/chishok/CST8703-Lab3.

Install Cheddar on **Windows**:

1. Download latest Windows zip file here
2. Extract zip file and run `cheddar.exe`

Or install on **Ubuntu**:

1. Run the script

   ```
   ./scripts/install_cheddar.bash
   ```

2. Run Cheddar

   ```
   run-cheddar
   ```

Cheddar - A Real-Time Scheduling Simulator

A simple example of a system scheduling will be analyzed using Cheddar. The following instructions only provide minimal information to run a scheduling analysis. For more details regarding the Cheddar application, follow the online tutorial steps which were used to develop the following procedures. We wish to determine the scheduling of a system of periodic tasks with these timing characteristics:

Table 1: Cheddar Example Task Set

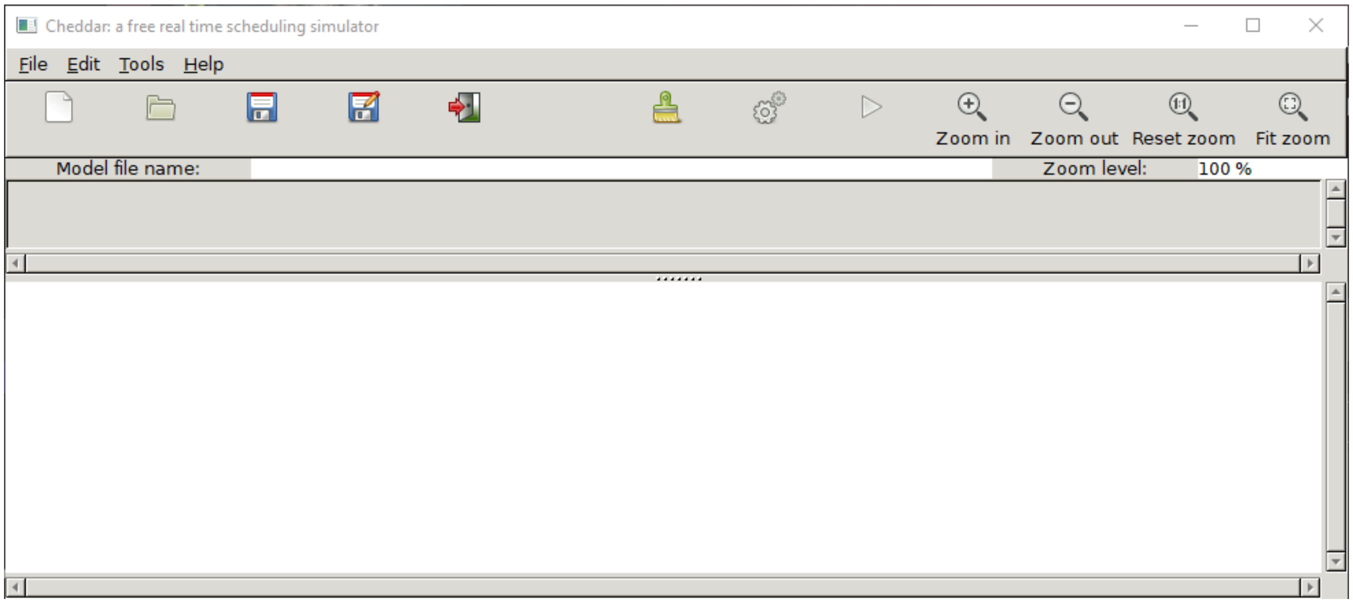
i	C_i	T_i	p_i
1	2	6	100
2	2	9	99
3	3	12	98

This system will be scheduled as rate monotonic (RM) where priority is higher for shorter interval durations T_i. Recall the symbol definitions:

Symbol	Definition
τ_i	Task
C_i	Worst-case execution time (WCET)
T_i	Interval deadline (period)
p_i	Priority

Perform the following steps to analyze the set of tasks:

1. The main cheddar window.

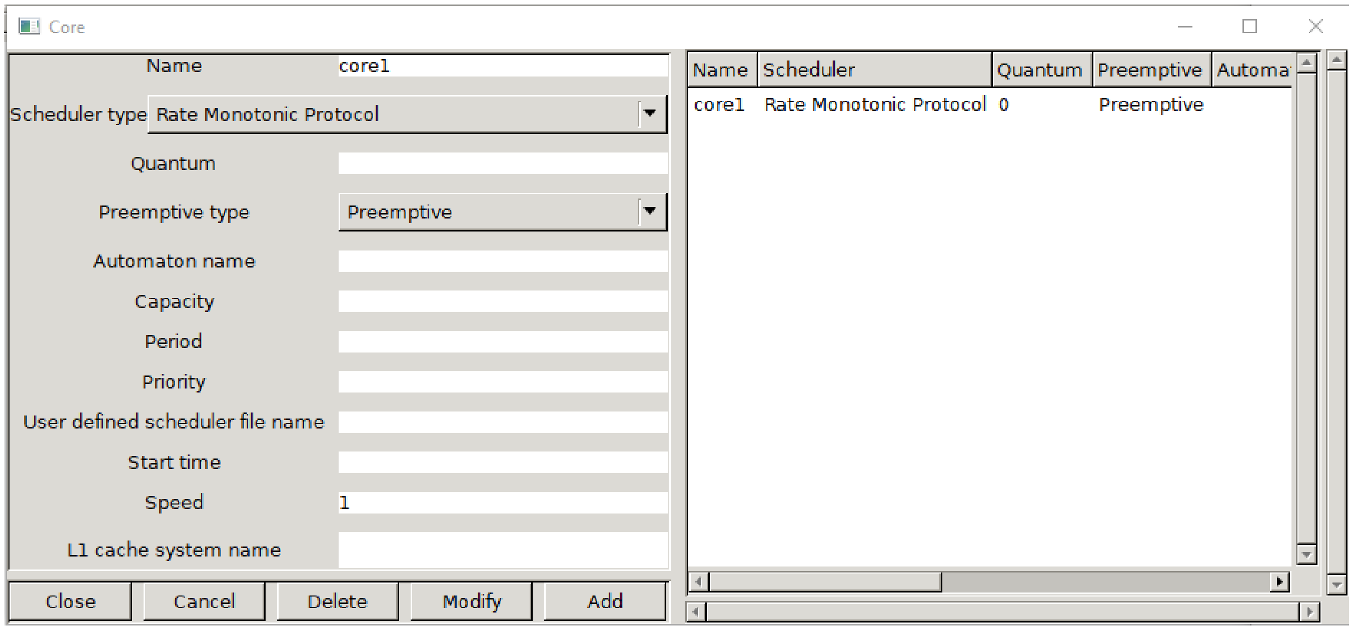


Model file name:

Zoom level: 100 %

Cheddar new window

2. **Add a `Core` component**



Name	core1
Scheduler type	Rate Monotonic Protocol
Quantum	
Preemptive type	Preemptive
Automaton name	
Capacity	
Period	
Priority	
User defined scheduler file name	
Start time	
Speed	1
L1 cache system name	

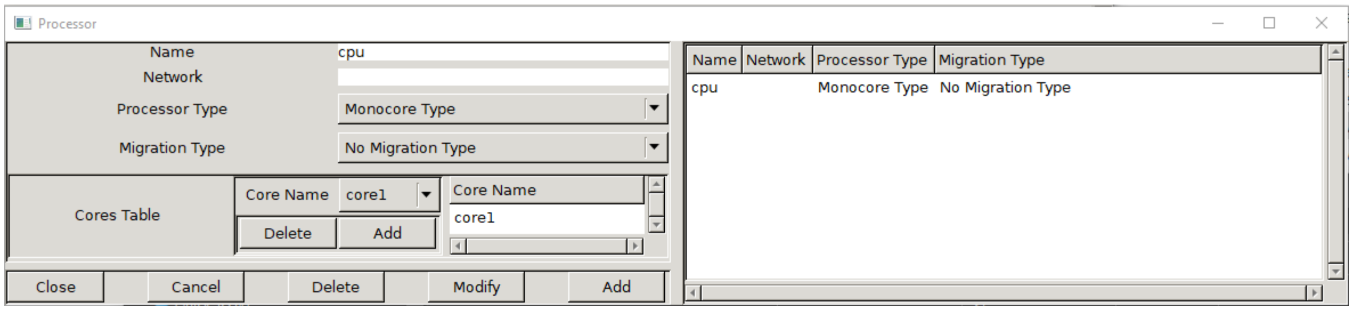
Name	Scheduler	Quantum	Preemptive	Automa
core1	Rate Monotonic Protocol	0	Preemptive	

Cheddar Core window

1. From the menu select `Edit -> Hardware -> Core`.
2. Fill out the `Name` as `core1`, select `Scheduler type` as `Rate Monotonic Protocol`, and `Preemptive type` as `Preemptive`.
3. Press the `Add` button.
4. Close the `Core` window.

3. **Define a `Processor` component**

1. From the menu select `Edit -> Hardware -> Processor`.



Name	cpu
Network	
Processor Type	Monocore Type
Migration Type	No Migration Type
Cores Table	Core Name core1

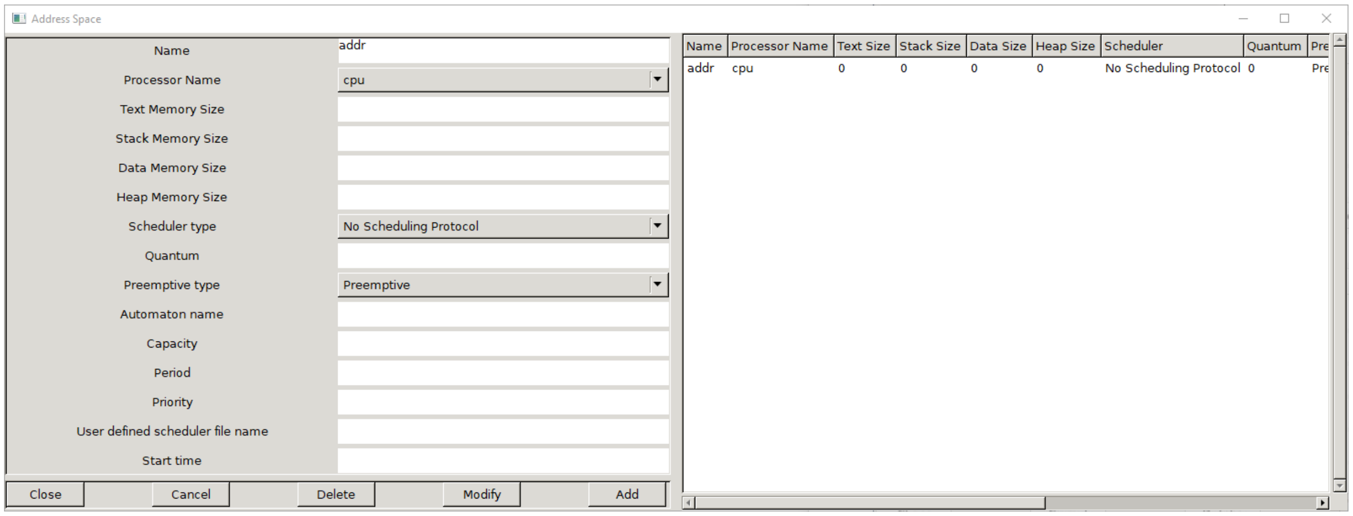
Core Name
core1

Name	Network	Processor Type	Migration Type
cpu		Monocore Type	No Migration Type

Cheddar Processor window

2. Fill out the Name as `cpu`.
3. In the `Cores Table` section set `Core Name` to `core1` and click on the core `Add` button just below. Press the `Add` button at the bottom row.
4. Close the Processor window.

4. **Define a software `Address Space`**



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

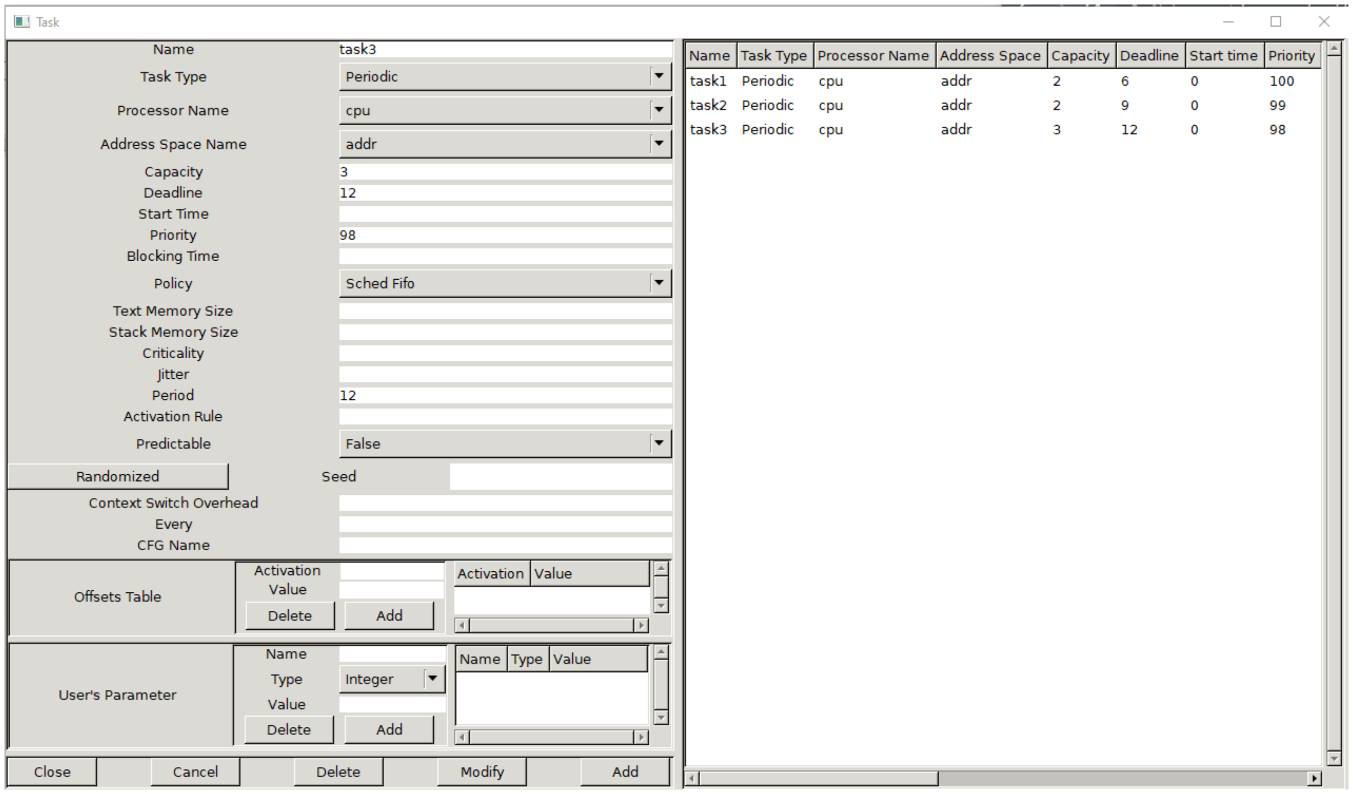
Name	Processor Name	Text Size	Stack Size	Data Size	Heap Size	Scheduler	Quantum	Pre
addr	cpu	0	0	0	0	No Scheduling Protocol	0	Pre

Cheddar Address Space window

1. From the menu select `Edit -> Software -> Address Space`.
2. Fill out the `Name` as `addr`.
3. Set `Processor Name` to `cpu` and click on the `Add` button.
4. Close the Address Space window.

5. **Add software `Task`**

1. From the menu select `Edit -> Software -> Task`.
2. Fill out the `Name` as `task1`.
3. Set `Task Type` to `Periodic`, `Processor Name` to `cpu`, `Address Space Name` to `addr`, and `Policy` to `Sched Fifo`.
4. From Table 1 above:
   1. Set the `Capacity` to the task 1 worst-case execution time ($C_1 = 2$ from table).
   2. Set the `Period` and `Deadline` to the period $T_1 = 6$.
   3. Set the `Priority` to $p_i = 100$.



Name	task3
Task Type	Periodic
Processor Name	cpu
Address Space Name	addr
Capacity	3
Deadline	12
Start Time	
Priority	98
Blocking Time	
Policy	Sched Fifo
Text Memory Size	
Stack Memory Size	
Criticality	
Jitter	
Period	12
Activation Rule	
Predictable	False
Seed	
Context Switch Overhead	
Every	
CFG Name	
Activation	
Value	
Name	
Type	Integer
Value	

Activation	Value

Name	Type	Value

Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Priority
task1	Periodic	cpu	addr	2	6	0	100
task2	Periodic	cpu	addr	2	9	0	99
task3	Periodic	cpu	addr	3	12	0	98

Figure 3: Cheddar task window

4. Click on the `Add` button.
5. Repeat previous step for tasks 2 and 3 with names `task2` and `task3` and corresponding WCET, period, and priorities from Table 1.
6. The task list should appear like the Cheddar task window screenshot in Figure 3

The model is now ready to run. Press the gears button to check if the model is feasible. A series of feasibility test are performed. In this case, two tests are run to verify that the task set is feasible with rate monotonic scheduling. For references, click on the menu `Help -> Scheduling References`. The feasibility result will be verified with our own implementation in the next section of this lab.

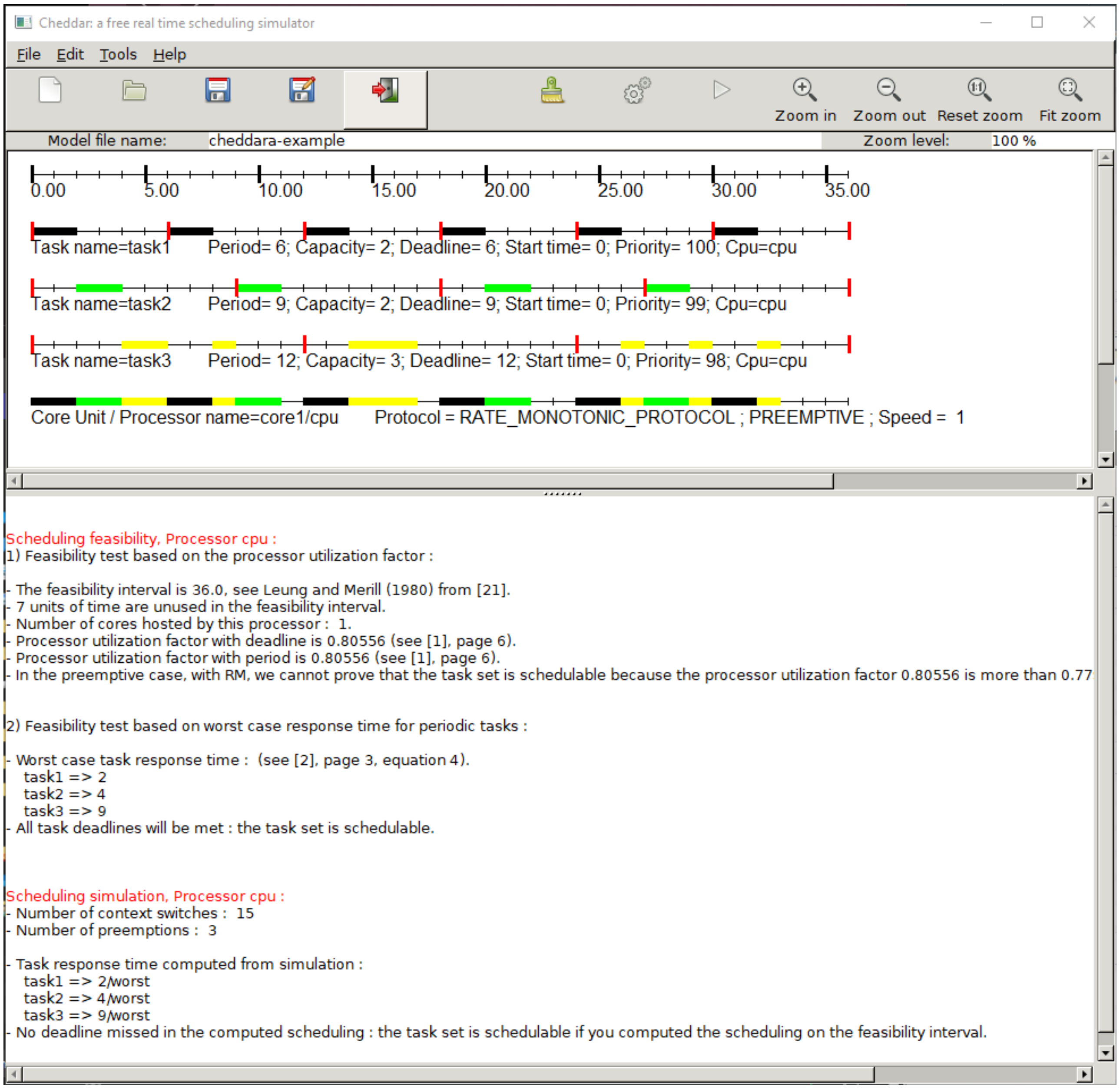
Depending on the scheduling policy and algorithm chosen, it may not be possible to determine feasibility without running a simulation. To run a simulation, press the `Play` button. Accept the default duration. The following figure shows the result of the feasibility tests and simulation with each task processing times and deadlines displayed until the entire sequence repeats:

The simulation duration is called the hyperperiod. If no deadline is missed in the simulation, the task set is schedulable.

Coding Task: Compare Rate Monotonic Feasibility With Cheddar Simulation

The feasibility test flowchart (Figure 2) will be implemented in code to determine the feasibility of scheduling a rate monotonic (RM) task set. We will run our feasibility tests and compare them with Cheddar.

In the file `FeasibleWorkload/src/main.c`, the first two task set examples are defined:


Cheddar: a free real time scheduling simulator
File Edit Tools Help
Zoom in
Zoom out
Reset zoom
Fit zoom
Model file name: cheddara-example
Zoom level: 100 %
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
Task name=task1 Period= 6; Capacity= 2; Deadline= 6; Start time= 0; Priority= 100; Cpu=cpu
Task name=task2 Period= 9; Capacity= 2; Deadline= 9; Start time= 0; Priority= 99; Cpu=cpu
Task name=task3 Period= 12; Capacity= 3; Deadline= 12; Start time= 0; Priority= 98; Cpu=cpu
Core Unit / Processor name=core1/cpu Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE ; Speed = 1

```
Scheduling feasibility, Processor cpu :
1) Feasibility test based on the processor utilization factor :

- The feasibility interval is 36.0, see Leung and Merill (1980) from [21].
- 7 units of time are unused in the feasibility interval.
- Number of cores hosted by this processor :  1.
- Processor utilization factor with deadline is 0.80556 (see [1], page 6).
- Processor utilization factor with period is 0.80556 (see [1], page 6).
- In the preemptive case, with RM, we cannot prove that the task set is schedulable because the processor utilization factor 0.80556 is more than 0.77


2) Feasibility test based on worst case response time for periodic tasks :

- Worst case task response time :  (see [2], page 3, equation 4).
   task1 => 2
   task2 => 4
   task3 => 9
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor cpu :
- Number of context switches :  15
- Number of preemptions :  3

- Task response time computed from simulation :
   task1 => 2/worst
   task2 => 4/worst
   task3 => 9/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Cheddar Scheduling Analysis Result.

Task Set ex0

i	C_i	T_i
1	2	6
2	2	9
3	3	12

Task Set ex1

i	C_i	T_i
1	2	6
2	2	9
3	3	12

> **NOTE**: Several task sets from this lab are from examples and exercises in [1] and may be used as a resource to verify the implementation (but is not required).

The priorities are not needed since these are assumed to be rate-monotonic (descending priority with ascending period). When the program `FeasibleWorkload` is run, the output provide the incorrect utilization factor and the tests are incorrectly identified as not feasible.

In order to fix the feasibility tests modify the file `FeasibleWorkload/src/ac_feasibility.c` between the comments

```
    /*
     * STUDENT WORK SECTION BEGIN
     * ===========================
     */
```

and

```
    /*
     * =========================
     * STUDENT WORK SECTION END
     */
```

Follow the instruction in the source carefully. Test the program by running and debugging `FeasibleWorkload`. Verify the output to look like this:

```
ex0:
  0:(C=1,T=2)  1:(C=1,T=10)  2:(C=2,T=15)
  Utilization factor U = 0.7333
  Least upper bound U_lub = 0.7798
  Feasible (U less than 1.0): Maybe
  Feasible (RM LUB): Yes
  Feasible (RM hyperbolic): Yes
  Feasible (DM response time): Yes

ex1:
```

```
0:(C=3,T=5)  1:(C=1,T=8)  2:(C=1,T=10)
Utilization factor U = 0.8250
Least upper bound U_lub = 0.7798
Feasible (U less than 1.0): Maybe
Feasible (RM LUB): Maybe
Feasible (RM hyperbolic): Yes
Feasible (DM response time): Yes
```

> **NOTE**: In the function `ac_feasibility_dm_response_time` in `FeasibleWorkload/src/ac_feasibility.c`, a bonus task is to implement the algorithm from Figure 1. A solution to this function will be presented in class as scheduled. You may attempt the function implementation before the solution is provided to obtain the bonus marks.

Now that the feasibility test is implemented correctly, modify `FeasibleWorkload/src/main.c` by adding the following task sets:

Task Set ex2

i	C_i	T_i
1	1	2
2	1	5
3	1	7
4	2	13

Task Set ex3

i	C_i	T_i
1	1	4
2	2	6
3	3	10

Task Set ex4

i	C_i	T_i
1	1	2
2	1	4
3	4	16

Task Set ex5

i	C_i	T_i
1	1	4
2	2	6

i	C_i	T_i
3	3	8

Modify the Cheddar model in Section with the task sets ex2, ex3, ex4, and ex5 above. Evaluate the feasiblity, run the simulation, and take a screenshot of the result for each task set.

Check your work by running:

```
./scripts/run_submission.sh
```

Expected contents of output.txt (some output hidden):

```
ex0:
  0:(C=1,T=2)  1:(C=1,T=10)  2:(C=2,T=15)
  Utilization factor U = 0.7333
  Least upper bound U_lub = 0.7798
  Feasible (U less than 1.0): Maybe
  Feasible (RM LUB): Yes
  Feasible (RM hyperbolic): Yes
  Feasible (DM response time): Yes

ex1:
  0:(C=3,T=5)  1:(C=1,T=8)  2:(C=1,T=10)
  Utilization factor U = 0.8250
  Least upper bound U_lub = 0.7798
  Feasible (U less than 1.0): Maybe
  Feasible (RM LUB): Maybe
  Feasible (RM hyperbolic): Yes
  Feasible (DM response time): Yes

ex2:
  0:(C=1,T=2)  1:(C=1,T=5)  2:(C=1,T=7)  3:(C=2,T=13)
  ...
  Feasible (DM response time): No

ex3:
  0:(C=1,T=4)  1:(C=2,T=6)  2:(C=3,T=10)
  ...
  Feasible (DM response time): Yes

ex4:
  0:(C=1,T=2)  1:(C=1,T=4)  2:(C=4,T=16)
  ...
  Feasible (DM response time): Yes

ex5:
  0:(C=1,T=4)  1:(C=2,T=6)  2:(C=3,T=8)
  ...
```

```
Feasible (DM response time): No
```

Analysis

1. Compare your results from running `FeasibleWorkload` compared to the Cheddar simulation. Are they all in agreement?
2. What are the hyperperiods of each task set calculated by the Cheddar simulations?
3. For the task set ex4, what do you notice about the Cheddar simulation time (hyperperiod) and the task 3 period? In a real-world implementation how confident would you be implementing the task set and guaranteeing deadlines are met? Refer to the simulation timing diagram to justify your answer.

Submission

Include your modified source code and a lab report. Include screenshots of Cheddar simulations. If you forked the repo on your personal GitHub, commit and push your changes, then download your repo as a zip file. If you do not use GitHub, compress the source code in a zip file or tar file. Be sure that the command `./scripts/run_submission.sh` produces the expected output.

References

[1] G. C. Buttazzo, *Hard real-time computing systems*, vol. 24. Springer US, 2011. doi: 10.1007/978-1-4614-0676-1.