



UNIVERSITÉ DE TECHNOLOGIE DE
COMPIÈGNE

MI11

Rapport de TP : Modélisation AADL et ordonnancement

Clément BLANQUET et Rafik CHENNOUF

Juin 2017

Sommaire

1	Exercice 1 : Ordonnancement Rate Monotonic	3
	Question 1	3
	Question 2	3
	Question 3	3
	Question 4	5
	Questions 5 et 6	7
2	Exercice 2 : Ordonnancement EDF	9
	Question 1	9
	Question 2	9
	Question 3	11
	Question 4	11
	Question 5	12
3	Exercice 3 : Moniteur médical multi-paramètres	16
	Question 1	16
	Question 2	16
	Question 3	18
	Question 4	19
	Question 5	20
	Question 6	21

Table des figures

1.1	Résultat de l'ordonnancement préemptif avec Cheddar - Ex1Q4 . .	5
1.2	Résultat de l'ordonnancement non préemptif avec Cheddar - Ex1Q4	6
1.3	Résultat de l'ordonnancement préemptif avec Cheddar - Ex1Q5 . .	7
1.4	Résultat de l'ordonnancement non préemptif avec Cheddar - Ex1Q5	8
2.1	Résultat de l'ordonnancement préemptif avec Cheddar - Ex2Q4 . .	11
2.2	Résultat de l'ordonnancement non préemptif avec Cheddar - Ex2Q4	12
2.3	Résultat de l'ordonnancement préemptif avec Cheddar - Ex2Q5 . .	15
3.1	Résultat de l'ordonnancement avec Cheddar	18
3.2	Tâches ordonnançables	18
3.3	Résultat de l'ordonnancement avec Cheddar avec deux processeurs .	19
3.4	Résultat de l'ordonnancement avec Cheddar avec deux processeurs sachant que la tâche getPA s'exécute sur b	21

Exercice 1 : Ordonnancement Rate Monotonic

Question 1

Le taux d'utilisation du processeur est :

$$\sum U_i = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} = \frac{7}{29} + \frac{1}{5} + \frac{2}{10} \simeq 0.64.$$

Sachant que l'on a trois tâches, la condition pour que le système soit ordonnançable est :

$$3 * (2^{1/3} - 1) > \sum U_i$$

$$3 * (2^{1/3} - 1) \simeq 0.78 > \sum U_i.$$

Le système est donc ordonnançable.

Question 2

Selon l'algorithme Rate Monotonic, la tâche la plus prioritaire est celle qui a la plus petite période. La tâche la plus prioritaire est donc la tâche T2 (P=5), suivi de T3 (P=10) puis T1 (P=29).

Question 3

```

1 thread T1
  end T1;

3
5 thread implementation T1.impl
  properties
    Dispatch_Protocol => Periodic ;
    Period => 29 ms
    Compute_Execution_Time => 1 ms .. 7 ms
    Deadline => 29 ms
    Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
11 end T1.Impl;
```

```

13 thread T2
14 end T2;
15
16 thread implementation T2.impl
17   properties
18     Dispatch_Protocol => Periodic ;
19     Period => 5 ms;
20     Compute_Execution_Time => 1 ms .. 1 ms
21     Deadline => 5 ms;
22     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
23 end T2.Impl;
24
25 thread T3
26 end T3;
27
28 thread implementation T3.impl
29   properties
30     Dispatch_Protocol => Periodic ;
31     Period => 10 ms;
32     Compute_Execution_Time => 1 ms .. 2 ms;
33     Deadline => 10 ms;
34     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
35 end T3.Impl;
36
37 process taches
38 end taches;
39
40 process implementation taches.impl
41   subcomponents
42     T1 : thread T1.impl;
43     T2 : thread T2.impl;
44     T3 : thread T3.impl;
45 end taches.impl;
46
47 processor cpu
48 end cpu;
49
50 processor implementation cpu.impl
51   properties
52     Scheduling_Protocol => RATE_MONOTONIC_PROTOCOL;
53     Cheddar_Properties::Preemptive_Scheduler => True ; --ou False
54 end cpu.impl;
55
56
57 system top
58 end top;
59
60 system implementation top.impl
61   subcomponents

```

```

63   cpu : processor cpu.impl;
      taches : process taches.impl;
      properties
65   Actual_Processor_Binding => reference cpu applies to taches;
end top.impl;

```

Question 4

Version préemptive :

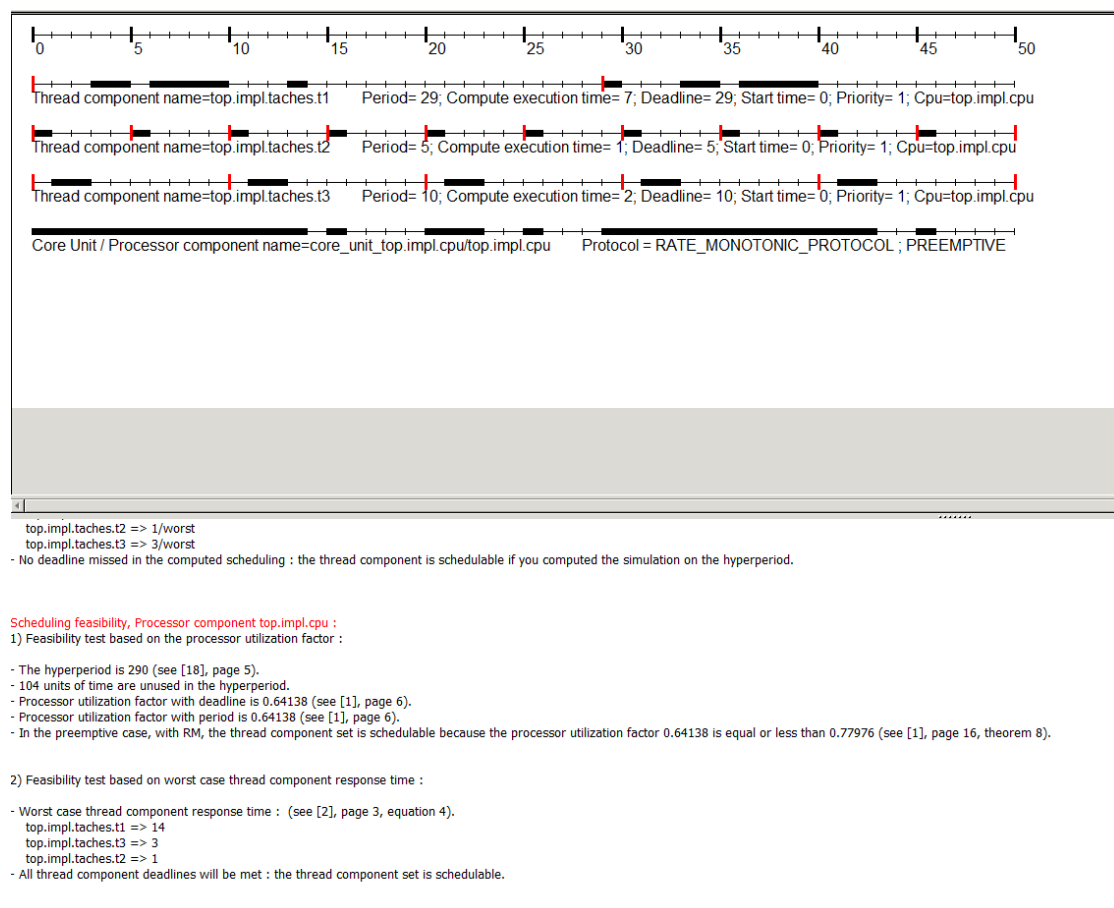


FIGURE 1.1 – Résultat de l’ordonnancement préemptif avec Cheddar - Ex1Q4

On observe sur la figure 1.1 que le système est bel est bien ordonnannçable comme nous l’avions indiqué à la première question. On remarque également qu’il

a besoin de 14 unités de temps pour que toutes les tâches se terminent au moins une fois.

Version non préemptive :

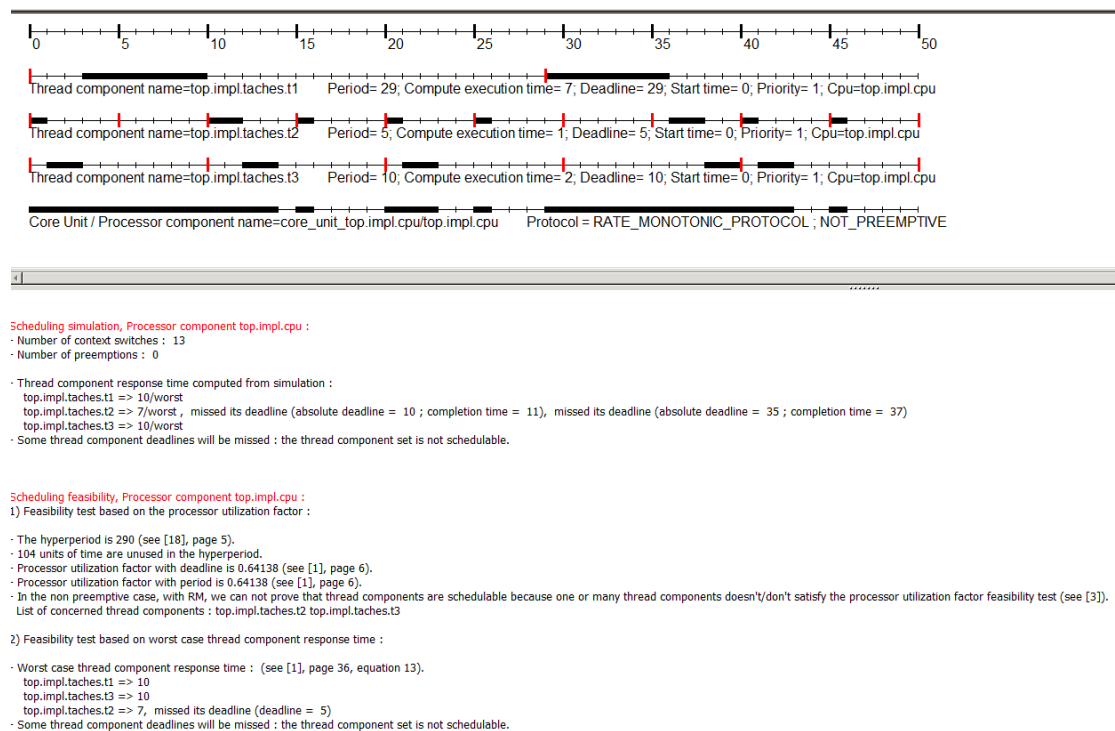
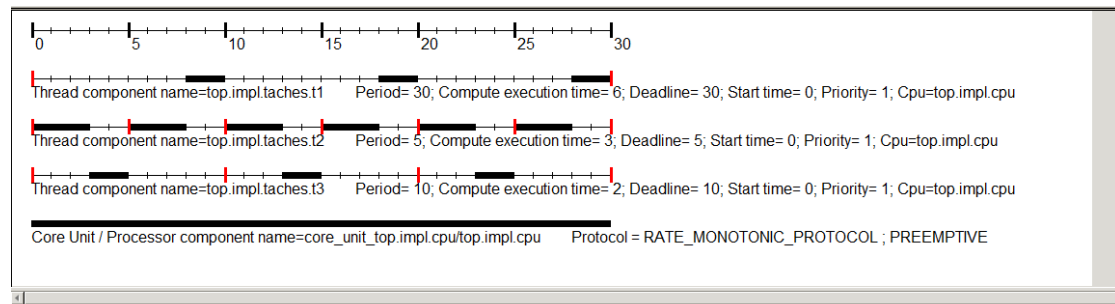


FIGURE 1.2 – Résultat de l'ordonnancement non préemptif avec Cheddar - Ex1Q4

On remarque sur la figure 1.2 que toutes les tâches ne respectent pas leur échéance en mode non préemptif. En effet, la tâche T2 manque sa deadline. Le système n'est donc pas ordonnançable en non préemptif.

Questions 5 et 6

Version préemptive :



Scheduling simulation, Processor component top.impl.cpu :

- Number of context switches : 11
- Number of preemptions : 2

- Thread component response time computed from simulation :

- top.impl.taches.t1 => 30/worst
- top.impl.taches.t2 => 3/worst
- top.impl.taches.t3 => 5/worst

- No deadline missed in the computed scheduling : the thread component is schedulable if you computed the simulation on the hyperperiod.

Scheduling feasibility, Processor component top.impl.cpu :

1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 30 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [11], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with RM, the thread component set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [19], page 13).

2) Feasibility test based on worst case thread component response time :

- Worst case thread component response time : (see [2], page 3, equation 4).
- top.impl.taches.t1 => 30
- top.impl.taches.t3 => 5
- top.impl.taches.t2 => 3
- All thread component deadlines will be met : the thread component set is schedulable.

FIGURE 1.3 – Résultat de l'ordonnancement préemptif avec Cheddar - Ex1Q5

En mode préemptif (figure 1.3), les deadlines sont toutes respectées mais on remarque que le système a besoin de 30 unités de temps pour que toutes les tâches se terminent au moins une fois (contre 14 à la question précédente). Le système est donc bien plus lent.

Version non préemptive :

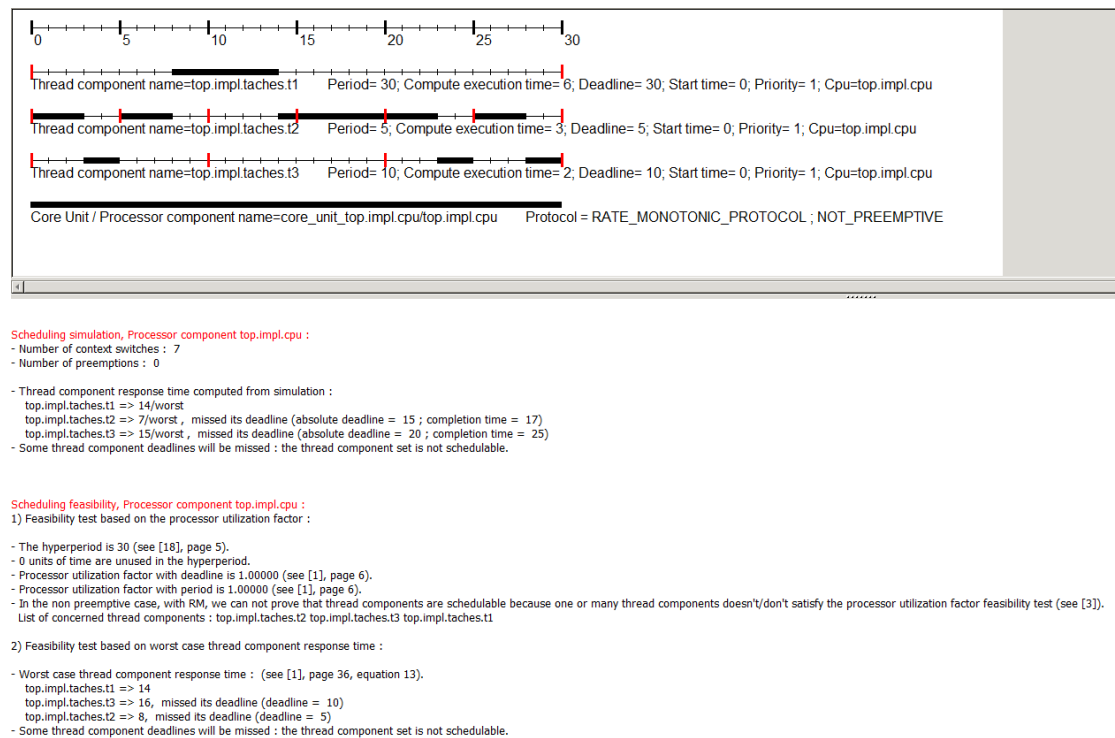


FIGURE 1.4 – Résultat de l’ordonnancement non préemptif avec Cheddar - Ex1Q5

En mode non préemptif, 2 tâches (T3 et T2) ratent leur deadlines. On en conclut que ces nouvelles données rendent le système bien moins efficace.

Exercice 2 : Ordonnancement EDF

Question 1

Le taux d'utilisation du processeur est :

$$\sum U_i = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} = \frac{5}{12} + \frac{2}{6} + \frac{5}{24} \simeq 0.96.$$

La condition pour que le système soit ordonnançable est :

$$\sum U_i < 1$$

Le système est donc ordonnançable.

Question 2

```

1 thread T1
2 end T1;

4 thread implementation T1.impl
5   properties
6     Dispatch_Protocol => Periodic ;
7     Period => 12 ms;
8     Compute_Execution_Time => 1 ms .. 5 ms;
9     Deadline => 12 ms;
10    Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
11 end T1.Impl;

12
13 thread T2
14 end T2;

16 thread implementation T2.impl
17   properties
18     Dispatch_Protocol => Periodic ;
19     Period => 6 ms;
20     Compute_Execution_Time => 1 ms .. 2 ms;
21     Deadline => 6 ms;
22    Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
23 end T2.Impl;
24
```

```
thread T3
26 end T3;

28 thread implementation T3.impl
    properties
30     Dispatch_Protocol => Periodic ;
    Period => 24 ms;
32     Compute_Execution_Time => 1 ms .. 5 ms;
    Deadline => 24 ms;
34     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
end T3.impl;
36

process taches
38 end taches;

40 process implementation taches.impl
    subcomponents
42     T1 : thread T1.impl;
    T2 : thread T2.impl;
44     T3 : thread T3.impl;
end taches.impl;
46

processor cpu
48 end cpu;

50 processor implementation cpu.impl
    properties
52     Scheduling_Protocol => EARLIEST_DEADLINE_FIRST_PROTOCOL;
    Cheddar_Properties::Preemptive_Scheduler => True ;
54 end cpu.impl;
56

system top
58 end top;

60 system implementation top.impl
    subcomponents
62     cpu : processor cpu.impl;
    taches : process taches.impl;
64     properties
    Actual_Processor_Binding => reference cpu applies to taches;
66 end top.impl;
```

Question 3

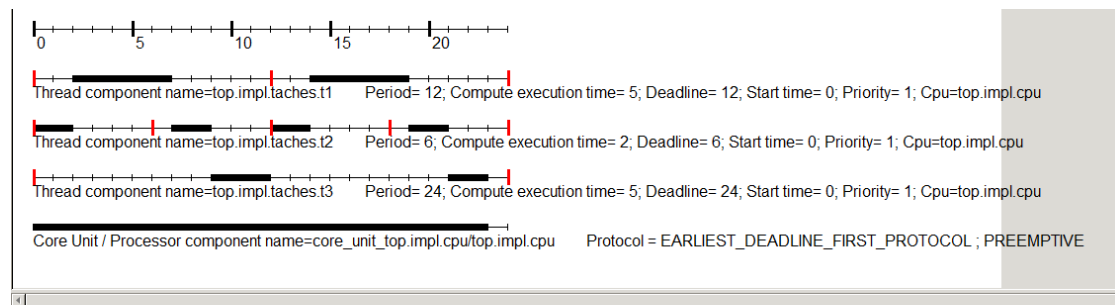
La période d'étude étant 24 (plus longue période), la quantité de temps libre peut être calculée grâce au taux d'utilisation du processeur calculé en question 1 :

$$(1 - \sum U_i) * 24 = 1.$$

Il y a donc une unité de temps libre sur la période d'étude.

Question 4

Version préemptive :



Scheduling simulation, Processor component top.impl.cpu :

- Number of context switches : 7
- Number of preemptions : 1

- Thread component response time computed from simulation :

top.impl.taches.t1 => 7/worst
top.impl.taches.t2 => 3/worst
top.impl.taches.t3 => 23/worst

- No deadline missed in the computed scheduling : the thread component is schedulable if you computed the simulation on the hyperperiod.

Scheduling feasibility, Processor component top.impl.cpu :

1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 24 (see [18], page 5).
- 1 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.95833 (see [1], page 6).
- Processor utilization factor with period is 0.95833 (see [1], page 6).
- In the preemptive case, with EDF, the thread component set is schedulable because the processor utilization factor 0.95833 is equal or less than 1.00000 (see [1], page 8, theorem 2).

2) Feasibility test based on worst case thread component response time :

- Worst case thread component response time :

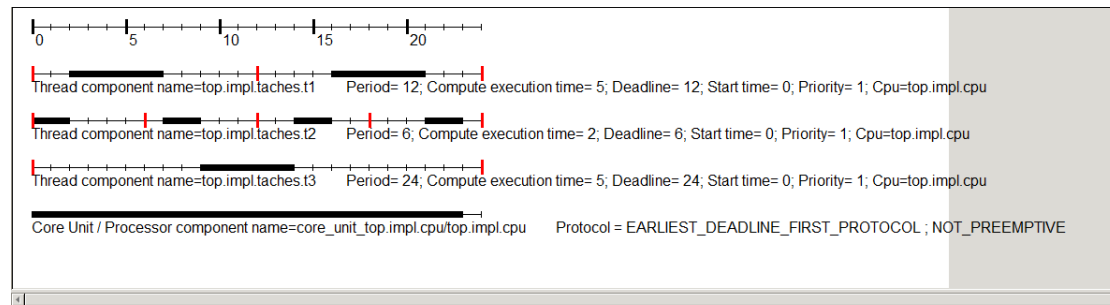
top.impl.taches.t1 => 11
top.impl.taches.t2 => 5
top.impl.taches.t3 => 23

- All thread component deadlines will be met : the thread component set is schedulable.

FIGURE 2.1 – Résultat de l'ordonnancement préemptif avec Cheddar - Ex2Q4

En mode préemptif (figure 2.1), toutes les tâches respectent leurs deadlines et on voit bien qu'il y a une unité de temps libre sur la période d'étude.

Version non préemptive :



Scheduling simulation, Processor component top.impl.cpu :

- Number of context switches : 6
- Number of preemptions : 0
- Thread component response time computed from simulation :
 - top.impl.taches.t1 => 9/worst
 - top.impl.taches.t2 => 5/worst
 - top.impl.taches.t3 => 14/worst
- No deadline missed in the computed scheduling : the thread component is schedulable if you computed the simulation on the hyperperiod.

Scheduling feasibility, Processor component top.impl.cpu :

- 1) Feasibility test based on the processor utilization factor :
 - The hyperperiod is 24 (see [18], page 5).
 - 1 units of time are unused in the hyperperiod.
 - Processor utilization factor with deadline is 0.95833 (see [1], page 6).
 - Processor utilization factor with period is 0.95833 (see [1], page 6).
 - In the non preemptive case, with EDF, the thread component set is not schedulable because one or many thread components doesn't/don't satisfy the processor utilization factor feasibility test (see [8]).
 - List of thread components not schedulable : top.impl.taches.t3
- 2) Feasibility test based on worst case thread component response time :
 - Worst case thread component response time :
 - top.impl.taches.t1 => 21, missed its deadline (deadline = 12)
 - top.impl.taches.t2 => 5
 - top.impl.taches.t3 => 42, missed its deadline (deadline = 24)
 - Some thread component deadlines will be missed : the thread component set is not schedulable.

FIGURE 2.2 – Résultat de l'ordonnancement non préemptif avec Cheddar - Ex2Q4

Sur la version non préemptive (figure 2.2), deux tâches ne respectent pas leurs échéances : T1 et T3.

Question 5

Nouveau code :

```

1 thread T1
2 end T1;

4 thread implementation T1.impl
5   properties
6     Dispatch_Protocol => Periodic ;
7     Period => 12 ms;
8     Compute_Execution_Time => 1 ms .. 5 ms;
9     Deadline => 12 ms;
10    Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;

```

```
end T1.Impl;  
12  
thread T2  
14 end T2;  
  
16 thread implementation T2.impl  
    properties  
18     Dispatch_Protocol => Periodic ;  
     Period => 6 ms;  
20     Compute_Execution_Time => 1 ms .. 2 ms;  
     Deadline => 6 ms;  
22     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;  
end T2.Impl;  
24  
thread T3  
26 end T3;  
  
28 thread implementation T3.impl  
    properties  
30     Dispatch_Protocol => Periodic ;  
     Period => 24 ms;  
32     Compute_Execution_Time => 1 ms .. 5 ms;  
     Deadline => 24 ms;  
34     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;  
end T3.Impl;  
36  
thread TA1  
38 end TA1;  
  
40 thread implementation TA1.impl  
    properties  
42     Dispatch_Protocol => Background ;  
     Compute_Execution_Time => 1 ms .. 1 ms;  
44     Cheddar_Properties::Dispatch_Absolute_Time => 7 ms;  
end TA1.Impl;  
46  
thread TA2  
48 end TA2;  
  
50 thread implementation TA2.impl  
    properties  
52     Dispatch_Protocol => Background ;  
     Compute_Execution_Time => 1 ms .. 3 ms;  
54     Cheddar_Properties::Dispatch_Absolute_Time => 12 ms;  
end TA2.Impl;  
56  
process taches  
58 end taches;
```

```
60 process implementation taches.impl
    subcomponents
62     T1 : thread T1.impl;
        T2 : thread T2.impl;
64     T3 : thread T3.impl;
        TA1 : thread TA1.impl;
66     TA2 : thread TA2.impl;
end taches.impl;

68
69 processor cpu
70 end cpu;

71
72 processor implementation cpu.impl
    properties
74     Scheduling_Protocol => EARLIEST_DEADLINE_FIRST_PROTOCOL;
        Cheddar_Properties::Preemptive_Scheduler => True ;
76 end cpu.impl;

77
78
79 system top
80 end top;

81
82 system implementation top.impl
    subcomponents
84     cpu : processor cpu.impl;
        taches : process taches.impl;
86     properties
        Actual_Processor_Binding => reference cpu applies to taches;
88 end top.impl;
```

Cela nous donne le résultat suivant avec Cheddar (en mode préemptif) :

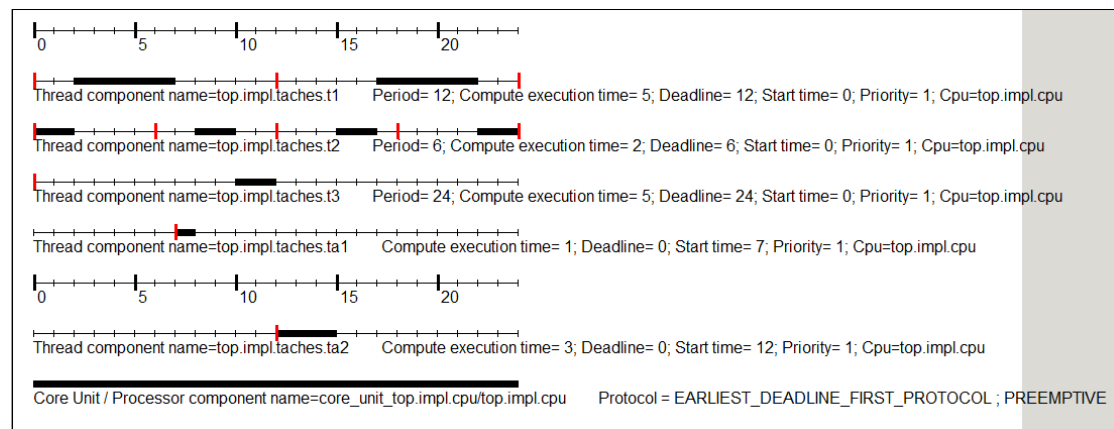


FIGURE 2.3 – Résultat de l’ordonnancement préemptif avec Cheddar - Ex2Q5

Ce n’est pas ordonnançable. En effet, il n’y avait qu’une unité de temps libre comme on l’a calculé à la question 3. Du coup, avec les tâches apériodiques qui préemptent les tâches périodiques, le système n’est plus ordonnançable (la tâche T3 ne respecte pas sa deadline).

Exercice 3 : Moniteur médical multi-paramètres

Question 1

On peut déterminer la priorité selon la plus petite période (Rate Monotonic), le plus petit temps de réponse dynamique (Earliest Deadline First) ou encore la plus petite laxité dynamique (Least Laxity First). On a décidé de choisir comme premier critère la période de la tâche (plus elle est petite, plus la tâche est prioritaire => Rate Monotonic) et en second critère, c'est à dire en cas d'égalité, la durée d'exécution de la tâche (plus elle est grande, plus la tâche est prioritaire).

Les paramètres et les priorités des différentes tâches à ordonnancer sont donnés dans le tableau ci-dessous :

Tâches	Période (ms)	Capacité (ms)	Priorité
getPA	10	1	3
getSO2	10	2	2
getECG	40	4	5
dpy	6	1	1
check	12	2	4

La tâche la plus prioritaire est la tâche **dpy** car elle a la période la plus petite (6 ms). Les tâches de priorité 2 et 3 sont respectivement les tâches **getSO2** et **getPA**. Ces deux tâches ayant la même période (10 ms), la tâche la plus prioritaire est celle avec une capacité supérieure, c'est à dire ici la tâche **getSO2**. La tâche de priorité 4 est la tâche **check** avec une période de 12 ms. Enfin, la tâche la moins prioritaire est la tâche **getECG** car elle possède la période la plus élevée.

Question 2

Nous avons implémenté les paramètres précédents dans le fichier AADL *health_monitor.aadl* comme ci-dessous :

```

— getPA
2 thread implementation read_sensor.PA
  properties
4   Dispatch_Protocol => Periodic ;
   Period => 10 ms;
6   Compute_Execution_Time => 1 ms .. 1 ms;
   Deadline => 10 ms;
8   Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
   Cheddar_Properties::Fixed_Priority => 253;
10 end read_sensor.PA;

— getECG
12 thread implementation read_sensor.ECG
  properties
14   Dispatch_Protocol => Periodic ;
   Period => 40 ms;
16   Compute_Execution_Time => 4 ms .. 4 ms;
   Deadline => 40 ms;
18   Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
   Cheddar_Properties::Fixed_Priority => 251;
20 end read_sensor.ECG;

— getSO2
22 thread implementation read_sensor.SO2
  properties
24   Dispatch_Protocol => Periodic ;
   Period => 10 ms;
26   Compute_Execution_Time => 2 ms .. 2 ms;
   Deadline => 10 ms;
28   Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
   Cheddar_Properties::Fixed_Priority => 254;
30 end read_sensor.SO2;

— check
32 thread implementation check_bounds.impl
  properties
34   Dispatch_Protocol => Periodic ;
   Period => 12 ms;
36   Compute_Execution_Time => 2 ms .. 2 ms;
   Deadline => 12 ms;
38   Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
   Cheddar_Properties::Fixed_Priority => 252;
40 end check_bounds.impl;

— dpy
42 thread implementation display.impl
  properties
44   Dispatch_Protocol => Periodic ;

```

```

50   Period => 6 ms;
    Compute_Execution_Time => 1 ms .. 1 ms;
    Deadline => 6 ms;
52   Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
    Cheddar_Properties::Fixed_Priority => 255;
54 end display.impl;

```

Question 3



FIGURE 3.1 – Résultat de l'ordonnancement avec Cheddar

- Worst case thread component response time : (see [2], page 3, equation 4).
 health_monitor.impl.appli.getcg => 17
 health_monitor.impl.appli.check => 6
 health_monitor.impl.appli.getpa => 4
 health_monitor.impl.appli.getso2 => 3
 health_monitor.impl.appli.dpy => 1
- All thread component deadlines will be met : the thread component set is schedulable.

FIGURE 3.2 – Tâches ordonnancables

On voit bien sur Cheddar que toutes nos tâches sont ordonnancables.

Question 4

En ajoutant un second processeur et en modifiant les paramètres des tâches en fonction de ce qui est donné dans le nouveau tableau, on remarque que les tâches ne respectent plus leurs contraintes temporelles. En effet, d'après Cheddar, la tâche **getPA** ne s'exécute plus du tout comme on peut le voir sur la copie d'écran suivante :

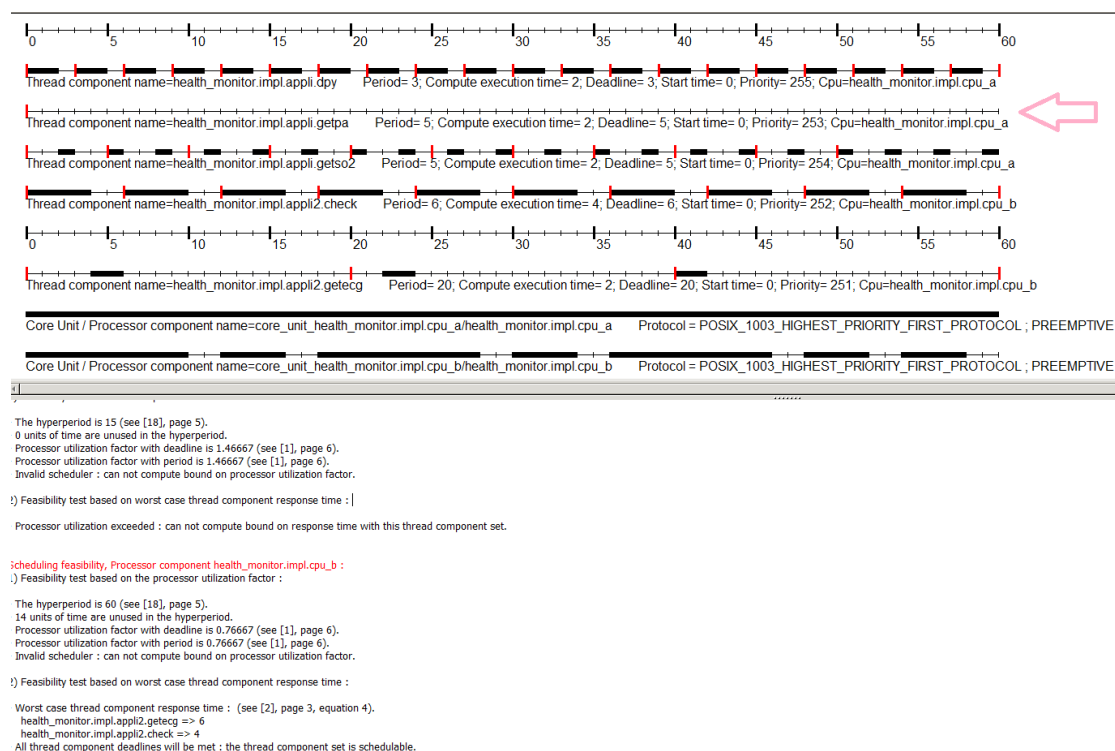


FIGURE 3.3 – Résultat de l'ordonnancement avec Cheddar avec deux processeurs

Ceci est dû au fait que les tâches **getPA** et **getSO2** ont exactement la même période et la même capacité. Comme nous avons décidé arbitrairement que la tâche **getSO2** est prioritaire sur la tâche **getPA**, cette tâche ne pourra jamais s'exécuter puisqu'à chaque fois que la tâche **dpy** aura terminé son exécution, c'est la tâche **getSO2** qui reprendra la main.

Question 5

Étant donné que la tâche **getPA** ne peut pas s'exécuter et que le processeur b est deux fois plus rapide que le processeur a, une solution simple serait de déplacer la tâche **getPA** vers le processeur b. Ceci permet d'alléger la charge de travail du processeur a et de permettre aux tâches **getPA** et **getSO2** d'être complètement indépendantes. Il faut aussi diviser la capacité initiale de la tâche **getPA** par 2 ce qui donne 1.

Tâches	Processeur	Capacité	Période
check	b	4	6
getECG	b	2	20
getPA	b	1	5
dpy	a	2	3
getSO2	a	2	5

Une deuxième solution aurait été de former un jeu de tâches harmoniques comme dans l'exercice 1. On cherche dans notre tableau des tâches qui possèdent des périodes multiples entre elles et on les regroupe sur le même processeur. Dans notre cas, on aurait pu regrouper les tâches **check** et **dpy** de périodes 6 et 3 sur le processeur b et les tâches **getPA**, **getSO2** et **getECG** de périodes 5, 5 et 20 sur le processeur a.

Question 6

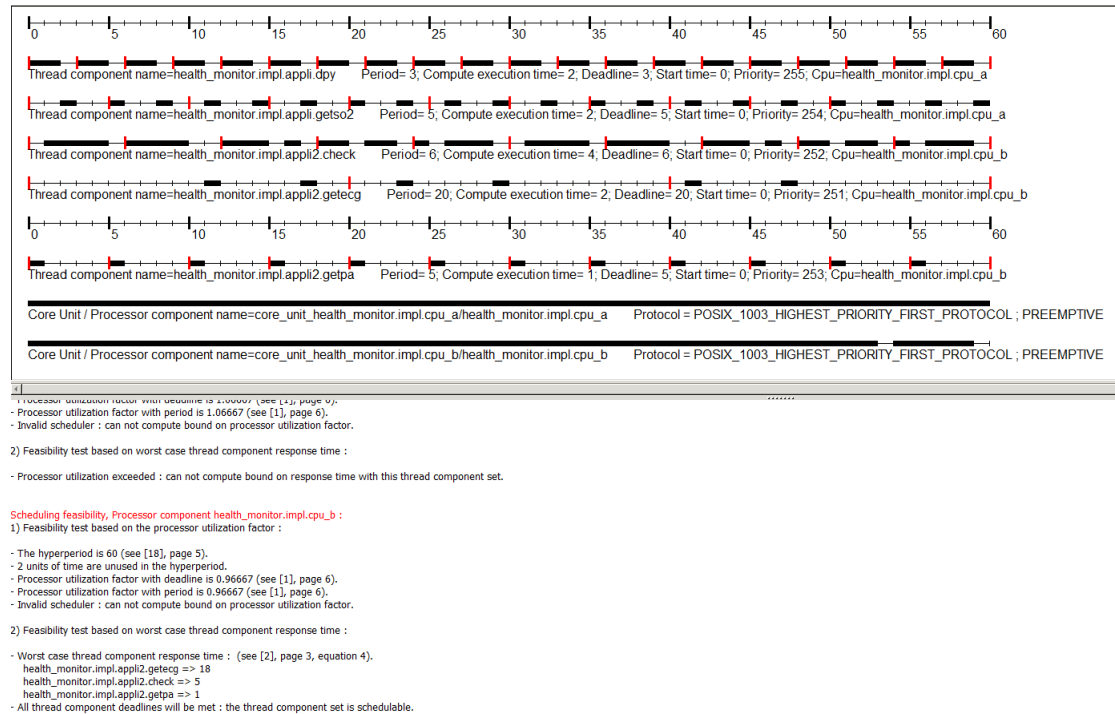


FIGURE 3.4 – Résultat de l’ordonnancement avec Cheddar avec deux processeurs sachant que la tâche **getPA** s’exécute sur b

On voit que cette solution permet à toutes les tâches de s’exécuter. Ci-dessous le code AADL complet :

```

1  —————
3  — Composants matériels
5  —————
7  — Bus
9  — Les capteurs sont sur bus CAN
11 bus CAN
13 end CAN;
15 — L’afficheur est sur bus LVDS
16 bus LVDS
17 end LVDS;

```

```

17 — Capteurs
18
19 — Capteur saturation O2
20 device SO2_sensor
21   features
22     SO2 : out data port;
23     iobus : requires bus access CAN;
24 end SO2_sensor;
25
26 device implementation SO2_sensor.impl
27 end SO2_sensor.impl;
28
29 — Capteur électrocardiogramme
30 device ECG_sensor
31   features
32     ECG : out data port;
33     iobus : requires bus access CAN;
34 end ECG_sensor;
35
36 device implementation ECG_sensor.impl
37 end ECG_sensor.impl;
38
39 — Capteur Pression artérielle
40 device PA_sensor
41   features
42     PA : out data port;
43     iobus : requires bus access CAN;
44 end PA_sensor;
45
46 device implementation PA_sensor.impl
47 end PA_sensor.impl;
48
49 — Autres périphériques
50 device LCD_display
51   features
52     fb : in data port;           — framebuffer
53     lvds : requires bus access LVDS; — lcd bus
54 end LCD_display;
55
56 device Alarm
57   features
58     trigger : in event data port;
59     iobus : requires bus access CAN;
60 end Alarm;
61
62 — Processeur
63 processor armadeus
64   features
65     iobus : requires bus access CAN;

```

```

65     lvds : requires bus access LVDS;
end armadeus;
67
processor implementation armadeus.impl
69   properties
     Scheduling_Protocol => POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL
     ;
71   Cheddar_Properties::Preemptive_Scheduler => True ;
end armadeus.impl;
73
processor armadeus2
75   features
     iobus : requires bus access CAN;
77     lvds : requires bus access LVDS;
end armadeus2;
79
processor implementation armadeus2.impl — NOUVEAU CPU
81   properties
     Scheduling_Protocol => POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL
     ;
83   Cheddar_Properties::Preemptive_Scheduler => True ;
end armadeus2.impl;
85


---


87 — Composants logiciels


---


89 thread read_sensor
     features
91     raw : in data port;
     cooked : out data port;
93 end read_sensor;

95 thread implementation read_sensor.PA
     properties
97     Dispatch_Protocol => Periodic ;
     Period => 5 ms;
99     Compute_Execution_Time => 1 ms .. 1 ms;
     Deadline => 5 ms;
101     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
     Cheddar_Properties::Fixed_Priority => 253;
103 end read_sensor.PA;

105 thread implementation read_sensor.ECG
     properties
107     Dispatch_Protocol => Periodic ;
     Period => 20 ms;
109     Compute_Execution_Time => 2 ms .. 2 ms;
     Deadline => 20 ms;
111     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;

```



```

113     Cheddar_Properties::Fixed_Priority => 251;
end read_sensor.ECG;

115 thread implementation read_sensor.SO2
    properties
117     Dispatch_Protocol => Periodic ;
    Period => 5 ms;
119     Compute_Execution_Time => 2 ms .. 2 ms;
    Deadline => 5 ms;
121     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
    Cheddar_Properties::Fixed_Priority => 254;
123 end read_sensor.SO2;

125 thread check_bounds
    features
127     PA : in data port;
    ECG : in data port;
129     SO2 : in data port;
    alarm : out event data port;
131 end check_bounds;

133 thread implementation check_bounds.impl
    properties
135     Dispatch_Protocol => Periodic ;
    Period => 6 ms;
137     Compute_Execution_Time => 4 ms .. 4 ms;
    Deadline => 6 ms;
139     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
    Cheddar_Properties::Fixed_Priority => 252;
141 end check_bounds.impl;

143 thread display
    features
145     PA : in data port;
    ECG : in data port;
147     SO2 : in data port;
    fb : out data port;
149 end display;

151 thread implementation display.impl
    properties
153     Dispatch_Protocol => Periodic ;
    Period => 3 ms;
155     Compute_Execution_Time => 2 ms .. 2 ms;
    Deadline => 3 ms;
157     Cheddar_Properties::Dispatch_Absolute_Time => 0 ms;
    Cheddar_Properties::Fixed_Priority => 255;
159 end display.impl;

```

```

161 process health_monitoring
    features
163     PA : in data port;
        ECG : in data port;
165     SO2 : in data port;
        fb : out data port;
167     alarm : out event data port;
end health_monitoring;
169

171 process implementation health_monitoring.impl
    subcomponents
173     getSO2 : thread read_sensor.SO2;
        dpy : thread display.impl;
175 end health_monitoring.impl;

177

179 process health_monitoring2
    features
        PA : in data port;
181     —ECG : in data port;
        SO2 : in data port;
183     fb : out data port;
        alarm : out event data port;
185 end health_monitoring2;

187

189 process implementation health_monitoring2.impl — NOUVELLE
    APPLICATION
    subcomponents
        getECG : thread read_sensor.ECG;
191     check : thread check_bounds.impl;
        getPA : thread read_sensor.PA;
193 end health_monitoring2.impl;

195
— système complet
197

199 system health_monitor
end health_monitor;

201 system implementation health_monitor.impl
    subcomponents
203     — Les bus
        can : bus CAN;
205     lvds : bus LVDS;

207     — Les processeurs
        cpu_a : processor armadeus.impl;

```

```

209     cpu_b : processor armadeus2.impl; — NOUVEAU CPU

211     — Les capteurs
    SO2_sense : device SO2_sensor;
213     PA_sense : device PA_sensor;
    ECG_sense : device ECG_sensor;
215
    — Les périphériques de sortie
217     lcd : device LCD_display;
    alarm : device Alarm;
219
    — L''application
221     appli : process health_monitoring.impl;
    appli2 : process health_monitoring2.impl; — NOUVELLE
    APPLICATION
223
225 connections
    — connexions de bus
227     bus access can -> SO2_sense.iobus;
    bus access can -> PA_sense.iobus;
229     bus access can -> ECG_sense.iobus;
    bus access can -> alarm.iobus;
231     bus access can -> cpu_a.iobus;

233     bus access lvds -> lcd.lvds;
    bus access lvds -> cpu_a.lvds;
235
    — connexions de ports
237     so2_conn : data port SO2_sense.SO2 -> appli.SO2;
    pa_conn : data port PA_sense.PA -> appli.PA;
239     ecg_conn : data port ECG_sense.ECG -> appli.ECG;
    fb_conn : data port appli.fb -> lcd.fb;
241     alarm_conn : event data port appli.alarm -> alarm.trigger;

243 properties
    — Allocation logiciel – plateforme d''exécution
245     Actual_Processor_Binding => reference cpu_a applies to appli;
    Actual_Processor_Binding => reference cpu_b applies to appli2;
    — NOUVEAU CPU

247     Actual_Connection_Binding => reference can applies to so2_conn;
249     Actual_Connection_Binding => reference can applies to pa_conn;
    Actual_Connection_Binding => reference can applies to ecg_conn;
251     Actual_Connection_Binding => reference can applies to alarm_conn;
    Actual_Connection_Binding => reference lvds applies to fb_conn;
253 end health_monitor.impl;

```