# TD Real-Time Systems (version 7.1)

## 1 RM, EDF and LLF (uni-processor)

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T).

T1: (C = 1, T = 3)
T2: (C = 1, T = 4)
T3: (C = 2, T = 6)

For each of the RM, EDF and LLF scheduling policies on a uni-processor:

• Calculate U, the processor utilization factor, what can we conclude from it?
• Give the chronogram of the tasks. What comment can we make?

Reminder: $n(2^{(1/n)}-1) = 0{,}78$ pour n = 3.

## 2 Response Time (uni-processor)

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T).

T1: (C = 25, T = 100)
T2: (C = 50, T = 200)
T3: (C = 100, T = 300)

Calculate their response time to determine if they are schedulable with RMS on a uni-processor.

Reminder: $R_i(t) = \Sigma_{j \text{ in } hp(i)} C_j * \lceil t/T_j \rceil$ with hp (i) containing the tasks with higher priority than task i

## 3 Aperiodic Task Servers

### *3.1 T1 being a periodic task*

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T).

T1: (C = 2, T = 10)
T2: (C = 4, T = 14)
T3: (C = 5, T = 14)

Demonstrate without doing the complete chronogram that the system is schedulable.

### *3.2 T1 being a deferred server*

We now assumed that T1 is a deferred server. We introduce in the previous system A, an aperiodic task characterized by a release time 28, a budget 6

Give the chronogram illustrating the scheduling of T1, T2, T3 and A from t = 25. Explain the issue et the reasons.

### *3.3 T1 being a polling server*

It is now assumed that T1 is a polling server. We introduce the same aperiodic task. Give the chronogram and the response time of A.

### *3.4 T1 sporadic server*

It is now assumed that T1 is a sporadic server. We introduce the same aperiodic task. Give the chronogram and the response time of A.

## 4 Integrated Modular Avionic

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T). They run on a partitioned ARINC 653 mono-processor platform.

| Name | Budget | Period / Deadline |
|---|---|---|
| T1 | 1 | 3 |
| T2 | 1 | 6 |
| T3 | 2 | 6 |
| T4 | 2 | 12 |

We want to apply preemptive Rate Monotonic scheduling. In case of tie, the lexicographical order is used.

### 4.1 *Question 1*

Explain why the schedulability test cannot conclude on the schedulability of this task set. Calculate the chronogram over the hyper-period.

### 4.2 *Question 2*

Tasks T1 and T3 have the same level of criticality A and T2 and T4 have the same level of criticality B. We have to group the tasks by level of criticality to form partitions. We keep the previous scheduling as it is. Define the Partition Windows to enforce the criticality requirements and calculate the number of partition switches.

### 4.3 *Question 3*

We want to reduce this number of switches. Suggest another configuration of partition windows so as to reduce the number of partition switches.

## 5 EDF and LLF in multi-processors systems (m cores)

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T).

$T_1$ : (C=2*epsilon, T)

…

$T_m$ : (C=2*epsilon, T)

$T_{m+1}$ : (C=T, T+epsilon)

### 5.1 *Utilization factor*

Calculate the utilization factor of the system when epsilon is small and T is large.

### 5.2 *Global EDF*

Apply Global EDF to this task set. What do we see?

### 5.3 *Global LLF*

Apply Global LLF to this set of tasks. What do we see?

### 5.4 *Conclusions*

What can we conclude from these examples?

## 6 Multi-processors partitioned scheduling (m cores)

Find a scheduling of **identical** synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T) so that they are not schedulable in partitioned but schedulable in global.

## 7 Global and partitioned schedulers on 2 cores

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T).

| Name | Budget | Period/Deadline |
|---|---|---|
| T1 | 4 | 6 |
| T2 | 7 | 12 |
| T3 | 4 | 12 |
| T4 | 10 | 24 |

### *7.1 Pfair*

Is the task system schedulable using PFair? Produce the chronogram (split into subtasks).

### *7.2 Global-RM*

Is the task system schedulable using Global-RM? Produce the chronogram.

### *7.3 Partitioned-RM*

Is the task system schedulable using a partitioned architecture, each partition scheduling its tasks with RM? Produce the chronogram.

## 8 Global vs partitioned scheduling

In this problem, we study the scheduling of a video surveillance application consisting of a set of tasks described below.

- The **Ta1** task takes care of the audio input device and manages the acquisition of audio samples. **Ta1 provides the inputs of the task Ta2 as an output**.
- The **Ta2** task manages the audio output device (speakers) and presents the audio stream on a supervision console. **Ta2 takes the outputs of Ta1 as input**.
- The **Tv1** and **Tv2** tasks perform similar functions for the video stream.

An engineer has to validate the temporal behaviour of this task set. It is further assumed that these are defined by the parameters below:

| Task | Budget | Period |
|---|---|---|
| Ta1 | 5 | 10 |
| Ta2 | 5 | 10 |
| Tv1 | 10 | 20 |
| Tv2 | 10 | 20 |

The deadlines are equal to the periods (tasks with implicit deadlines). The date of first activation of all tasks is the same and is zero (synchronous tasks).

Assume a **preemptive fixed priority scheduler** having 100 priority levels (from 0 to 99, the priority level 99 being the highest priority level).

### *8.1* Priorities to make tasks independent

The tasks are not independent (see description above). The task **Ta2** (respectively **Tv2**) must be activated on termination of the task **Ta1** (respectively **Tv1**).

From **Ta1** and **Ta2** (respectively **Tv1** and **Tv2**), which task must have the highest priority in order to ensure the precedence constraints expressed above.

**Once these constraints on priorities have been expressed, tasks Ta1 and Ta2 (resp. Tv1 and Tv2) will be considered as independent. They are activated at the beginning of their period ($a_i$=0) but their assignment of priority must guarantee that the precedencies are respected.**

## *8.2* Partitioned scheduling (independent tasks)

The execution platform consists of two identical processors. As a first approach, we want to apply a partitioned scheduling: the tasks **Ta1** and **Ta2** are placed on a first processor and the other tasks on the second. We use the properties on the priorities expressed in **question 8.1**. Calculate the scheduling over the hyper-period. Is the system schedulable?

## *8.3* Global scheduling (independent tasks)

In a second approach, a preemptive global scheduling with static priority is used to execute these tasks on the 2 processors, the allocation of priorities to tasks being to be determined. It is assumed that the migration of tasks can take place at any time. In the following, it is assumed that this scheduling makes it possible to correctly execute the set of tasks **Ta1**, **Ta2**, **Tv1** and **Tv2** while ensuring the properties on the priorities expressed in **question 4.1** in order to make these tasks independent.

Given the capabilities of **Tv1** and **Tv2** demonstrate that they necessarily consume a full processor over the entire hyper-period.

## *8.4* Property on priorities of Tv1 and Ta1

Prove at time t = 0, **Tv1** and **Ta1** must have the 2 highest priorities for the system to run properly.

## *8.5* Property on priorities of Tv2 and Ta2

Prove that one must have priority (**Tv2**) < priority (**Ta2**) for the system to run correctly at t = 5.

## *8.6* Opposite property on priorities of Tv2 and Ta2

Prove that one must have priority (**Tv2**) > priority (**Ta2**) for the system to run correctly after t = 10.

## *8.7* Comparison of approaches on a particular scenario

For this particular scenario, which approach (global or partitioned) would you recommend?

# 9 MC/DC code coverage

We have the following code

F3(x, y, z) : if (((x==1) && (y==2)) || (z==3)) {F1(x, y, z) ;} ; F2(x, y, z) ;

## *9.1 2 tests*

According to MC / DC, are the F3 (1, 2, 3) and F3 (2, 2, 3) tests redundant or complementary? Justify your answer.

## *9.2 Logical table*

Build the logical table.

# 10 CAN Bus

Recall the arbitration principles of the CAN bus. You detail the result of the simultaneous transmission of messages on the bus by tasks with identifiers 4, 5, 6 and 7.

## 11 Multi-processors scheduling

In this exercise, we schedule on two processors the following system of periodic synchronous tasks with implicit deadlines:

| Task | Budget | Period |
|---|---|---|
| T1 | 1 | 4 |
| T2 | 1 | 5 |
| T3 | 18 | 20 |

Question 1. How does Global- EDF work? Is this system schedulable using Global-EDF?

Question 2. We increase the period of T1 from 4 to 5. Is this system schedulable using Global-EDF? Comment on the results of the two questions?

## 12 Scheduling with blocking time

We want to execute on a mono-processor a set of tasks sharing 2 locks S1 and S2.

| Name | C | T | Execution code |
|---|---|---|---|
| T1 | 2 | 8 | L(S1) ; Exec(1) ; U(S1) ; Exec(1) ; |
| T2 | 3 | 10 | L(S1) ; Exec(1) ; U(S1) ; Exec(1) ; L(S2) ; Exec(1) ; U(S2) |
| T3 | 3 | 20 | Exec(1) ; L(S2) ; Exec(1) ; U(S2) ; Exec(1) ; |
| T4 | 7 | 40 | L(S1) ; Exec(3) ; U(S1) ; Exec(1) ; L(S2) ; Exec(3) ; U(S2) |

Let be the synchronous periodic tasks (ready at t = 0) with implicit deadlines (D = T). They run on a partitioned ARINC 653 mono-processor platform. We want to apply a pre-emptive Earliest Deadline First scheduling. We are studying the scheduling in the case of PIP-EDF and SRP-EDF synchronization protocols. Determine the blocking times. Apply system scheduling tests in the presence of blockages for EDF.

# 13 Energy-aware scheduling

In this exercise, we consider the following system of periodic synchronous tasks with implicit deadlines running on a mono-core processor. The priority of tasks is pre-established (see below) and the higher the priority value, the higher the task priority.

| **Tasks** | **Budget (HIE)** | **Budget (LOE)** | **Period** | **Priority** |
|---|---|---|---|---|
| T1 | 4 | 5 | 10 | 3 |
| T2 | 3 | 0 | 15 | 2 |
| T3 | 4 | 5 | 15 | 1 |
| T4 | 4 | 5 | 30 | 0 |

Our system is powered by a photovoltaic panel so that the system has to change behaviour when its energy level is below a given threshold. When the energy level is above the threshold (HIgh-Energy mode), the system runs at normal processor frequency and the scheduler executes 4 tasks with the allocated budgets given in the HIE column. When the energy level reaches the threshold (LOw-Energy mode), the system will reduce the processor frequency by 25%. As a consequence, the WCET and the task budget will increase and task T2 will no longer execute to compensate for the loss of computing power. In this case, the scheduler executes only 3 tasks (T2 is no longer running) with the allocated budget given in the LOE column.

To refine the system model, the scheduler will detect the frequency change when a task has exhausted its budget without completing. For instance, when the scheduler detects that task T2 has not completed after 3 units of execution, it will conclude that a frequency change happened, and will shift from scheduling 4 tasks with HIE budgets to scheduling 3 tasks with LOE budgets without restarting the tasks. Only task T2 will stop, while tasks T1, T3 and T4 are allowed to execute for one extra unit (+25%).

## *13.1* Schedulabity in HIE and LOE modes (case 1)

In this question, no frequency change happens. Provide the timeline (chronogram) in HIE mode (which includes T1, T2, T3 and T4 tasks) without changing frequency. Then provide the timeline in LOE mode (which includes only T1, T3 and T4 tasks). In both cases, we assume that all tasks execute for their maximum budget. Is the system schedulable in both modes? Use the timelines given below if needed.

## *13.2* Frequency change (case 1)

In this question, a frequency change happens while executing task T2. After executing task 2 for 3 units, the scheduler detects that task T2 has not completed. We assume that all tasks execute for their maximum budget. Provide the timeline when the frequency change is detected at the end of task T2 because of the exceeded budget. Is the system schedulable in this case?

## *13.3* Schedulabity in HIE and LOE modes (case 2)

We now assume task T2 is has priority 1 and task T3 has priority 2. In this question, no frequency change happens. In both cases, we assume all tasks execute for their maximum budget. Provide the timeline in HIE mode (which therefore includes T1, T2, T3 and T4 tasks) without changing frequency. Then provide the timeline in LOE mode (which includes only T1, T3 and T4 tasks). Is the system schedulable in both modes?

### *13.4* Frequency change (case 2)

In this question, the frequency change happens while executing task T2. After executing task 2 for 3 units, the scheduler detects that task T2 has not completed. We assume that all tasks execute for their maximum budget. Provide the timeline when the frequency change is detected at the end of task T2 because of the exceeded budget. Is the system schedulable in this case?

# SOLUTIONS

## 1 RM, EDF and LLF

U=1/3+1/4+2/6=11/12

The system can be scheduled by LLF and EDF because it checks the necessary and sufficient condition $U \leq 1$.

Nothing can be said for RMS because the sufficient condition is not verified because $U \geq n(2^{(1/n)}-1) = 0,78$ pour n = 3. On the other hand, the chronogram over hyper-period 12 shows that the system is schedulable with RMS.

## 2 Response Time

We calculate by iteration the response time given by $R_i(n+1)\ \Sigma_{j\leq i} C_j * \lceil R_i(n)/T_j \rceil$

for all tasks j of higher priority than task i. Knowing that $R_i(0) = C_i$. The final response time must be less than the deadline.

$R_1(0) = 25, R_1(1) = R_1(0) = 25$ *or* $R_1(1) < D_1$ : T1 respects its deadline with RMS.

$R_2(0) = 50, R_2(1) = 25*\lceil 50/100 \rceil + 50 = 75, R_2(2) = R_2(1) = 75, R_2(1) < D_2$ : T2 respects its deadline.

$R_3(0) = 100, R_3(1) = 25*\lceil 100/100 \rceil + 50*\lceil 100/200 \rceil + 100 = 175, R_3(1) = 25*\lceil 175/100 \rceil + 50*\lceil 175/200 \rceil + 100 = 200, R_3(1) = 25*\lceil 200/100 \rceil + 50*\lceil 200/200 \rceil + 100 = 200\ R_3(3) = R_3(2) = 200, R_3(3) < D_3$ : T3 respects its deadline.

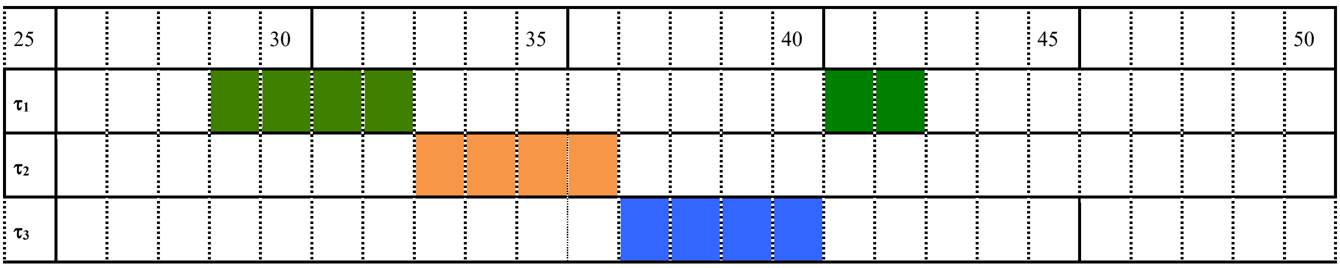
## 3 Aperiodic Task Servers

### *3.1 T1 being a periodic task*

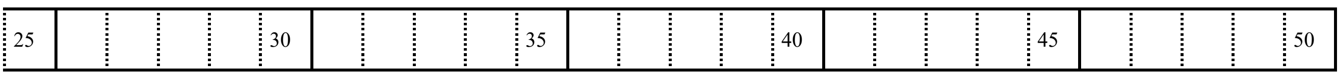
We calculate by iteration the response times to show that the system is schedulable. Note that we can regroup T2 and T3 since they have the same period.

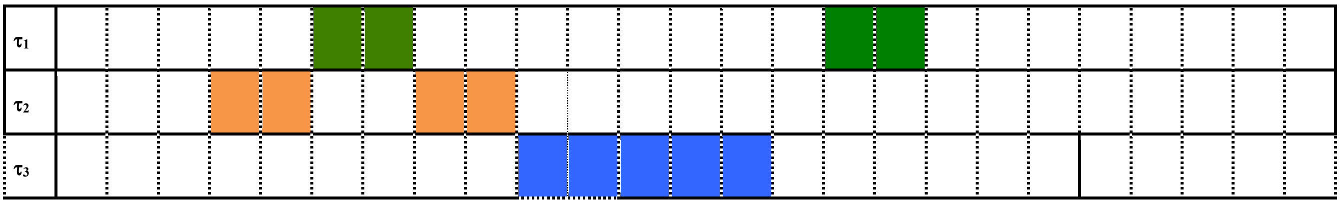
### *3.2 T1 being a deferred server*

Task T3 misses its deadline at T = 42.

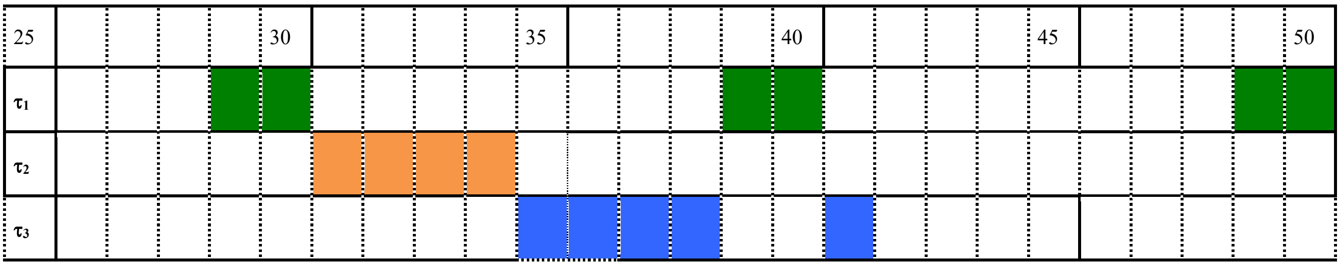



### *3.3 T1 being a polling server*






### 3.4 T1 being sporadic server




# 4 Integrated Modular Avionic

### 4.1 Question 1

U = 1 therefore with RMS we cannot conclude.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 | T2 | T3 | T1 | T3 | T4 | T1 | T2 | T3 | T1 | T3 | T4 |

### 4.2 Question 2

There are 8 partition switches (including the one occurring a the hyper period)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P2 | P1 | P1 | P1 | P2 | P1 | P2 | P1 | P1 | P1 | P2 |
| T1 | T2 | T3 | T1 | T3 | T4 | T1 | T2 | T3 | T1 | T3 | T4 |

### 4.3 Question 3

We group tasks T1 and T3 then T2 and T4. There are 3 partition switches.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P1 | P1 | P1 | P2 | P2 | P2 | P2 | P1 | P1 | P1 | P1 |
| T1 | T3 | T3 | T1 | T2 | T4 | T4 | T2 | T1 | T1 | T3 | T3 |

# 5 EDF and LLF in multi-processes systems (m cores)

### 5.1 Utilization factor

U = (2m/P)*epsilon + P/P+epsilon so U is approximately 1. Much less than m.

### 5.2 Global-EDF

With Global-EDF, T1 to Tm start on C1 to Cm cores. At t = 2 * epsilon, T1 to Tm end and Tm + 1 can run, ends at 2 * epsilon + P and miss its deadline.

### 5.3 Global-LLF

With Global-LLF, Tm + 1 starts on C1 and T1 to Tm-1 starts on the other cores then when T1 ends on C2, Tm starts on C2 and all the tasks respect their deadline.

### *5.4 Conclusions*

On the one hand, even if U = 1 Global-EDF can fail in multi-core systems (U=m) ! On the other hand, Global-LLF seems to dominate Global-EDF (proved in the littérature).

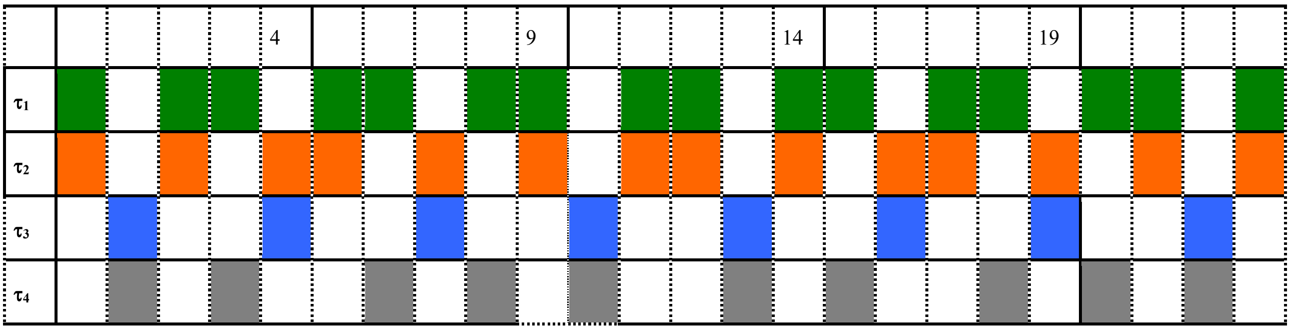
## 6 Multi-processors partitioned scheduling (m cores)

The idea is to ensure that the use of two (identical) tasks does not fit on one core (U> 1) and that the sum of the utilization factor fits on m cores. To do this, we choose m + 1 tasks of C = (T + epsilon) / 2.
2U = (T + epsilon) / T> 1. But (m + 1) (T + epsilon) / 2T <= m
if (T + epsilon) <= m (T-epsilon), so as soon as 2 <= m.

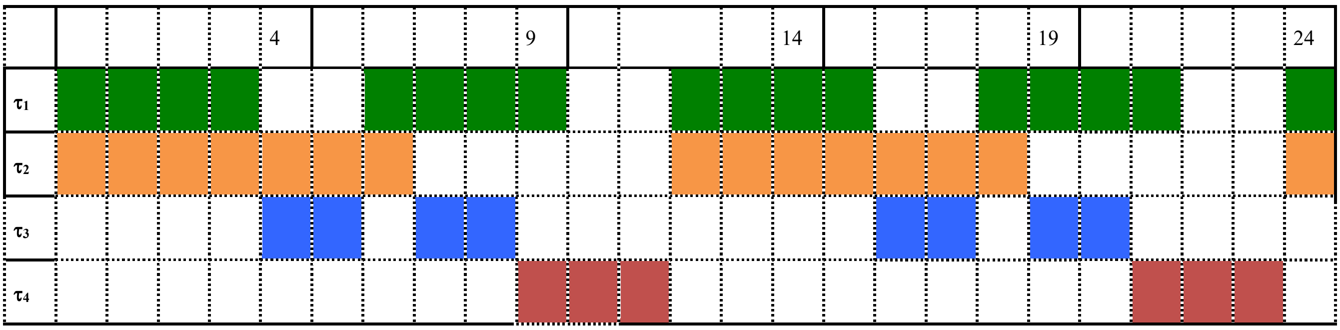
## 7 Global and partitioned schedulers on 2 cores

### *7.1 PFair*

For each task i, we create Ci sub-tasks (a unit j for each budget unit) such that the activation (resp the deadline) is $\lfloor (j-1) / Ui \rfloor$ (resp $\lceil j / Ui \rceil$) .
For T1, intervals are (0 ; 2), (1 ; 3), (3 ; 5), (4 ; 6), (6 ; 8), (7 ; 9), (9 ; 11), (10 ; 12) (U = 4/6)
For T2, intervals are (0 ; 2), (1 ; 4), (3 ; 6), (5 ; 7), (6 ; 9) (8.11), (10.12) (U = 7 / 12)
For T3, intervals are (0 ; 3), (3 ; 6), (6 ; 9), (9 ; 12) (U = 4/12)
For T4, intervals are (0 ; 3), (2 ; 5), (4 ; 8), (7 ; 10), (9 ; 12), (12 ; 15), (14 ; 17), (16 ; 20), (19 ; 22) (21 ; 24). (U = 10/24)




### *7.2 Global-RM*




T4's deadline miss at t=24.

### *7.3 Partitioned-RM*

The global utilisation factor being 2 for 2 cores available, each core must support the execution of tasks with a utilisation factor of 1. The only solution is to put T2 (7/12) with T4 (10/24) and therefore T1 (4/6) with T3 (4/12). We quickly calculate that the response time of T3 is 12 less than its deadline and that of T4 is 24.

# 8 Global vs partitioned scheduling

## *8.1* Priorities to make tasks independent

Tv1 (Ta1) have to start first so it has to be assigned a priority higher than Tv2 (Ta2).

## *8.2* Partitioned scheduling (independent tasks)

The hyper-period is 20 and as Ta1 (Tv1) has a highest priority compared to Ta2 (Tv2) it is first executed on core C1 (C2) during [0,5[ ([0,10[) and then Ta2 (Tv2) is executed, the overall system is schedulable.

## *8.3* Global scheduling (independent tasks)

U=2 and we have 2 cores.

## *8.4* Property on priorities of Tv1 and Ta1

At time t = 0, all tasks are ready (independent tasks). To enforce the right precedencies, the scheduler has to select Tv1 and Ta1 so they must have the 2 highest priorities.

## *8.5* Property on priorities of Tv2 and Ta2

At time t = 5, tasks Tv1, Ta2 and Tv2 are ready. Tv1 has a highest priority and is scheduled on C1. We cannot schedule Tv2 on C2 once Ta1 has completed for two reasons: Ta2 would miss its deadline and Tv2 would start before the completion of Tv1. Thus, priority (Ta2) > priority (Tv2).

## *8.6* Opposite property on priorities of Tv2 and Ta2

At time t = 10, tasks Ta1, Ta2 and Tv2 are ready. Ta1 has a highest priority and is scheduled on C1. We cannot schedule Ta2 on C2 once Tv1 has completed for two reasons: Tv2 would miss its deadline and Ta2 would start before the completion of Ta1.Thus, priority (Tv2) > priority (Ta2).

## *8.7* Comparison of approaches on a particular scenario

We prove that priority (Ta2) > priority (Tv2) and priority (Tv2) > priority (Ta2). Thus, we cannot schedule this system with a global scheduler with static priority.

# 9 MC/DC code coverage

## *9.1 Two tests*

They are redundant because like z == 3 one varies x and y without modifying the decision. The final test is still true.

## *9.2 Logical table*

| x==1 | y==2 | z==3 | decision |
|---|---|---|---|
| F | V | V | V |
| F | V | F | F |
| V | V | F | V |
| V | F | F | F |

# 10 CAN Bus

The bus starts transmitting the identifier over the bus, bit by bit. Bit 1 is recessive (the emitter loses in a conflict that would involve at least one node sending a bit 0). In the example, we will detail, bit by bit, the decisions that the nodes take to determine whether they are emitters or receivers (4 = 1000, 5 = 1001, 6 = 1010, 7 = 1011). Ultimately, nodes with weaker identifiers have the highest priority.

## 11 Multi-processors scheduling

Question 1: By doing the chronogram over the hyper-period of 20, we find that the system is schedulable.

Question 2: By doing the chronogram over the hyper-period of 20, we find that the system is not schedulable. We observe a scheduling anomaly.

## 12 Scheduling with blocking time

See example in slides.