

Real Time Operating System
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 04
Cyclic Scheduler

Welcome back. We have been looking at the real time tasks scheduling and we said that we will start with the simplest schedulers and then as we proceed we will look at more sophisticated tasks schedulers and different overrating systems depending on their sophistication they use different types of schedulers. And we said that the clock driven schedulers are the simplest scheduler.

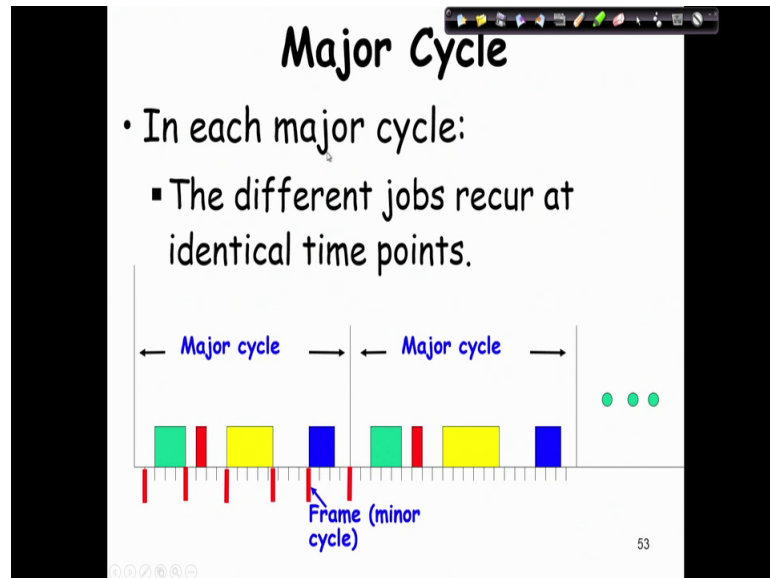
In the last lecture we started looking at simple cyclic executives and then we looked at the table driven scheduler and then we had started looking at the cyclic scheduler. We had mentioned that in the table driven scheduler the operating system maintains a table of the tasks and their time of execution and it sets a timer each time; the table driven scheduler is that requires setting of timers one set timers and number of times and this is a major overhead because we are talking of simple operating system requiring only one millisecond or less for each scheduler execution, each instance of execution of scheduler at the scheduling points.

And then we has said that the cyclic schedulers overcome this problem and also these are much more sophisticated in the sense that they can handled multi rate tasks, tasks requiring execution in different time periods and that to efficiently they require setting a cyclic periodic timer only once during the system initialization. But setting up the schedule for a cyclic scheduler is nontrivial compared to a table driven scheduler or a simple cyclic executive and we had said that if there n tasks with different periods then the period for which the schedule needs to be developed is given by the LCM of their periods.

During this LCM some tasks many repeat multiple number of times and some task may just occur only once. So, this LCM of the period is called as the hyper period or the major cycle and then the major cycle is divided into frames equal sized frames. And the periodics timer is set to give an interrupt at these frame boundaries and each time a periodic timer interrupt comes this scheduler runs and decides the next task to execute by

consulting the schedule table. Setting up the frame and assigning the tasks to the frames is a major challenge in this cyclic scheduling. Now, let us proceed looking at how are these frames decided frame duration and how are tasks assigned to the frames.

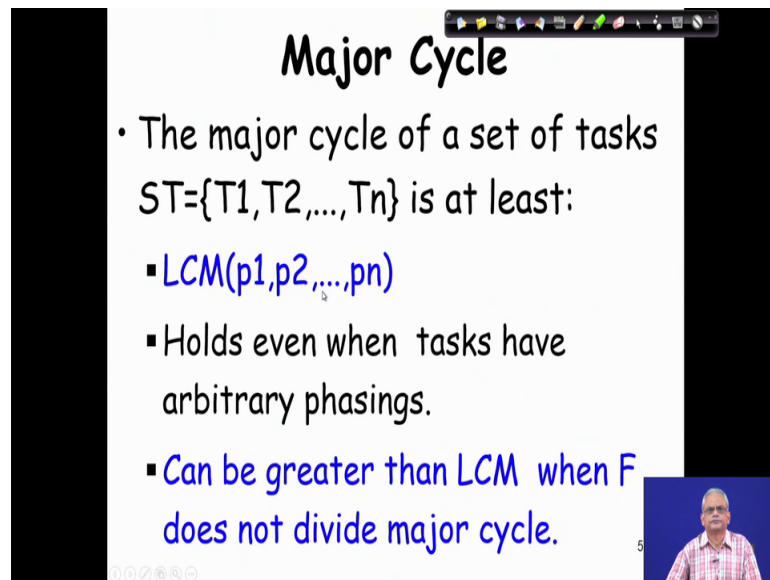
(Refer Slide Time: 04:14)



In the major cycle the tasks are assigned to frames let us say this green task to this first frame a red one to the second, yellow one to the third and so on and in the next major cycle they are repeated identically. So, basically the cyclic scheduler stores schedule for one major cycle which may have some number of frames may be let us say 12 frames and then the scheduled table will have entry 12 entries and each entry will identify which task or which program to run tasks and basically programs in these simple schedulers.

So, the first one program 1, program 2, next etcetera are to be run.

(Refer Slide Time: 05:21)



Major Cycle

- The major cycle of a set of tasks $ST=\{T1,T2,\dots,Tn\}$ is at least:
 - $LCM(p1,p2,\dots,pn)$
 - Holds even when tasks have arbitrary phasings.
 - Can be greater than LCM when F does not divide major cycle.


So, the major cycle is decided by the LCM of the task periods and even when the tasks are arbitrary phasings still the cyclic scheduler can be handled this because for few initial cycles until the phase time, the frame assigned to the task will remain unutilized because the task instances will start arising after the phase is over.

But what if the frame size does not divide the major cycle? We said that the LCM contains an integral number of frames or in other words the frame divides the major cycle. If you does not the frame does not divide the major cycle then we have some frame which does not complete at the major cycle boundary and therefore, the schedule has to be stored much larger than the major cycle and that is the reason why one of the requirement for setting of the frame size is that the frame size must divide the major cycle otherwise the schedule length to be stored in the table will be much larger.

(Refer Slide Time: 06:54)

Minor Cycle (Frame)

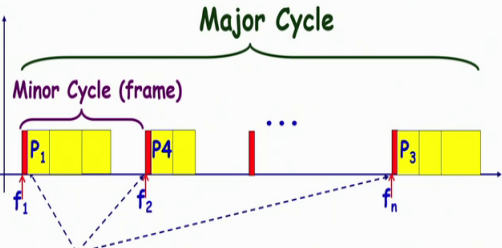
- Each major cycle:
 - Usually contains an integral number of minor cycles (frames).
- Frame boundaries are marked:
 - Through interrupts generated from a periodic timer.



So, we can assume that in a typical application will have the minor cycle are frames squarely dividing the major cycle. And the frame boundaries are marked by the interrupts generated by the periodic time.

(Refer Slide Time: 07:15)

Major and Minor Cycle



- Cyclic scheduler runs in response to a tick event
- Red bar shows time to execute scheduler

So, this is the major cycle which is decided best in the task periods is the LCM of the task periods and then that is divided into n number of frames all frames are equally equal size and then as the periodic scheduler is set to give a interrupt after every frame length as soon as the time the interrupt from the periodic timer comes the cyclic schedulers

starts running consults the schedule table and executes the corresponding task that is mentioned in the table. And the task may complete early that returns idol and then the next frame boundary occurs given by a interrupt from the periodic timer, the cyclic scheduler becomes active, consults the schedule table, finds the task to rerun next which is p 4 in this case and then initiates execution it completes early and then the next frame ends here. And after the major cycle this schedule is repeated and see here this small length of the red line shows the executing time indicative execution of the scheduler execution takes very small time compared to the task executing times.

So, these are efficient scheduler. In each schedule the work required is small as soon as the interrupt occurs just consults the schedule table and start the execution of the corresponding task there is no other activity, no timer setting nothing because it is a periodic timer it set once during the system initialization time.

(Refer Slide Time: 09:45)

Frame	Task
F1	T2
F2	T3
F3	T2

So, typical schedule table would look like the different frames and the tasks that are to be executed. Of course, we have shown this for our understanding, but then in a real task table we need not write the frame number etcetera. So, you just store the sequence in which task to be executed. So, just name of the programs executables that to be executed and the scheduler just indexes in this array of and the vector of tasks and then finds out the current one initiates execution and increments the entry to the table.

(Refer Slide Time: 10:36)

Constructing a Schedule

- Construct static schedule for a **Major Cycle**
- Cyclic Executive repeats this schedule
- There may be resulting idle intervals
 - if so attempt to arrange so they occur periodically

Cyclic Schedule

f_1 tasks ₁
...
f_5 tasks ₅

Task = (r, e)

T_1	=	(4, 1)
T_2	=	(5, 1.5)
T_3	=	(20, 1)
T_4	=	(20, 2)
H	=	20

Major Cycle (Hyperperiod)

We have already mentioned that one complication in using a cyclic scheduler is constructing a schedule. Two important aspects here one is how to fix the frame duration the second is about assigning tasks to the frames. So, after doing both of these actions we will have the schedule table where we have the frames and the tasks. So, let us look at the integrity of how to develop the schedule table.

(Refer Slide Time: 11:23)

Partitioning A Major Cycle into Frames

- Design steps:
 1. choose frame size,
 2. partition jobs into slices (if needed),
 3. place jobs/slices into frames.
- At Frames boundaries :
 - Cyclic executive performs scheduling
- There is no preemption within frame

Major Cycle (Hyperperiod)

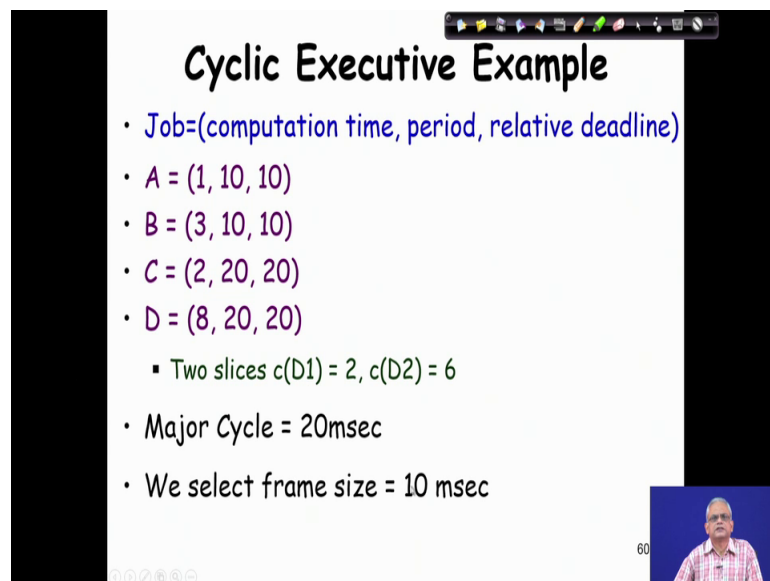
So, the steps that we need to do to develop the schedule the first thing to do is to choose the frame size and then we may not be always successful in the first attempt to get

acceptable frame size. If we get acceptable frame size then the job is rather easy the next step would be to place the jobs into the frames.

But often we will see that no frame size is suitable in that case you need to break some of the jobs into smaller jobs because as you will see through the examples, and the methodology is that the jobs which takes large execution time they are the culprits which header finding a proper frame size these make it difficult for setting the schedule and we may not be able to find the correct frame size.

And remember that these are simple scheduler these are not preemptive only the start execution of a program in each frame and it is not that some high priority task can arise and then we have to preempt some tasks etcetera nothing. Here we have a simple table here and the interrupts corresponding to the frame boundaries and the cyclic executive simple consults the schedule table and just starts execution of the corresponding task.

(Refer Slide Time: 13:26)



Cyclic Executive Example

- Job=(computation time, period, relative deadline)
- A = (1, 10, 10)
- B = (3, 10, 10)
- C = (2, 20, 20)
- D = (8, 20, 20)
 - Two slices $c(D1) = 2, c(D2) = 6$
- Major Cycle = 20msec
- We select frame size = 10 msec

60

Let us look at one example here. This we can do even without much background. So, I have given this simple example here. So, you have four tasks A B C D and each task sorry each job the task instance is defined by the computation time required. So, it A requires 1 millisecond, B requires 3 millisecond, C requires 2 and D requires 8 millisecond. The period is 10 for the first two, 20 for the next two and the deadline is same as the period. So, before the next instance next job arrives for this task the task

must execute. So, is D is a large task and we divide into 2 and 6 right now it may not be clear why 2 and 6 we will see that.

So, here the major cycle is 20 which is LCM of the period of these four jobs which is 20 millisecond and let us say we want to select a frame size each frame size must squarely divide the major cycle and therefore, we decide to choose table. But then why not 2 why not 5 as we proceed we will see that each frame must be able to accommodate at least I mean any job. So, any job should be able to run to completion in one of the frames.

So, the possible frame sizes are 1 2 5 10 and 20 and 1 2 5 cannot be used because we have a job here requiring 8. So, we can choose 10.

(Refer Slide Time: 16:02)

Schedule Example

1	4	6	8	10	11	14	20
A	B	C	D1	idle	A	B	D2

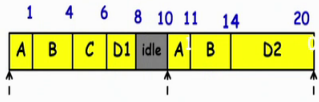
- Could we create a different schedule?
- Is it better to distribute the idle time?

61

So, then once having chosen that we can place these in the different frames, but let us look at more insights into how to do that.

(Refer Slide Time: 16:16)

Schedule Example



```
if Frame_no = 0 → run A ; B ; C ; D1;
else → run A ; B ; D2;
Frame_no = (Frame_no+1) mod 2;
```

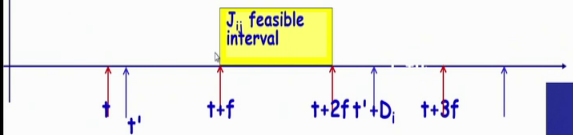
62

So, in this case the cycling scheduler is simple if it is frame number 0 there are two frame if we select 10 it runs A B C D otherwise it runs A B D and then the frame number is implemented.

(Refer Slide Time: 16:35)

Frame Size Constraints

- Every job needs to start and complete within a frame
 - $f \geq \max(T_i), 1 \leq i \leq n$
- Frame size f divides H (the Hyperperiod)
- Between the release time and deadline of every job there is at least one frame



62

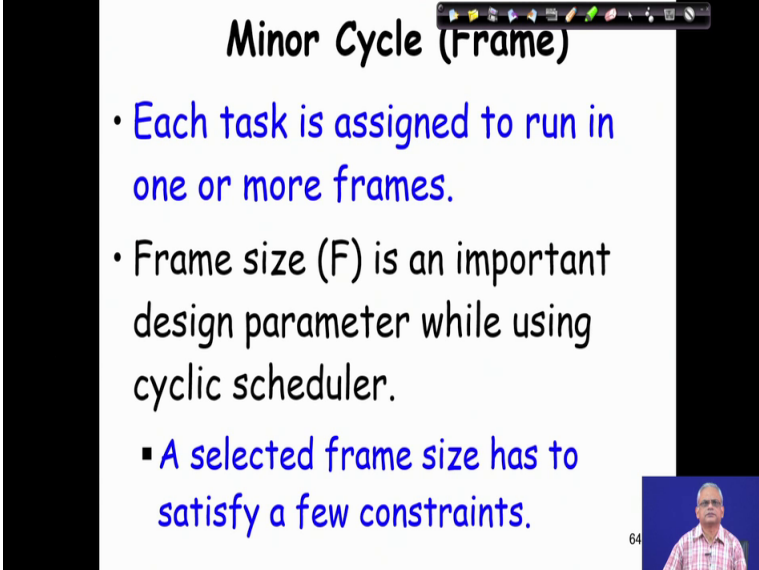
Now, let us see what are the constraints in choosing the frame size. The first thing is that the frame size must be larger than any task execution time because each task must be assigned to one frame and it must complete its execution in the frame. So, that is sets the lower bound on what can be frame size. So, the maximum execution time of a task that is

the lower bound for the frame we cannot make the frame any smaller than that. The second constraint in the frame size is that the frame size must divide the major cycle or the hyper period, and the third constraint that we would need is that between the release time of a task and the deadline of the task there must be at least one frame.

So, the time at which the task recurs. So, this is the release time and the deadline there must exist at least one frame. To think of it why this necessary is that if the task arrives the job the task instance and the job arrives or is released after the frame boundary then we cannot assign that to that frame because at the frame boundary the scheduler must initiate the execution of the job.

So, if the job arrives after the frame boundary we cannot execute the initiation of the job in this frame and the deadline if it is somewhere here or within this frame then of course, that job cannot complete execution before the deadline and that is the reason why we need at least one clear frame to exist between the release of a task of any task before it at the time it is released and its time of completion the deadline there must exist a clear frame. To explain this further I just want to take some examples and with some diagrams I will explain that further that why we need one frame one clear frame to exist between the release time and the completion time.

(Refer Slide Time: 19:30)



Minor Cycle (Frame)

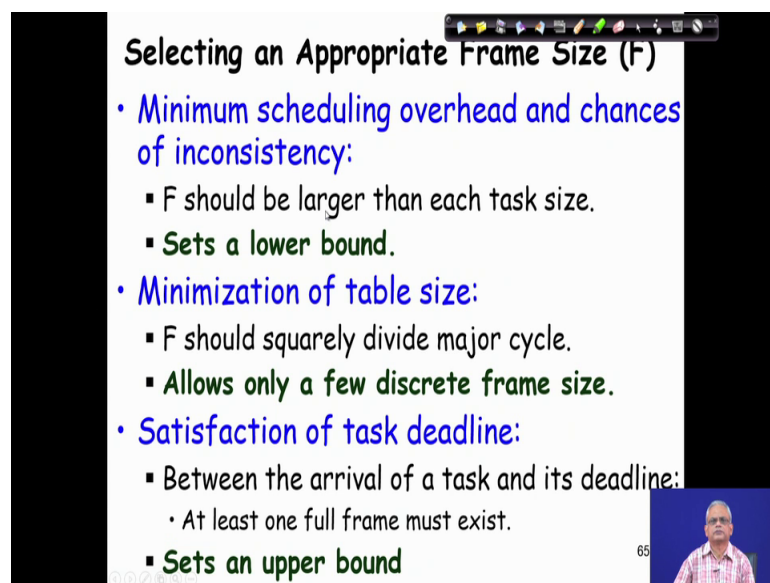
- Each task is assigned to run in one or more frames.
- Frame size (F) is an important design parameter while using cyclic scheduler.
 - A selected frame size has to satisfy a few constraints.

But before that the first thing is that the frame size is an important design parameter and there are three constraints that the frame size must satisfy, one is that it must squarely

divide the major cycle, the second is that the frame size must be larger than the largest job size, the third is that between the release of every task and its corresponding deadline there must exist one frame.

The first two are easy to check that whether the frame size divides the major cycle squarely or not, the second one is also easy that the frame must be larger than the largest task size, but it is the third constraint which requires us to do some thinking that there must be a clear frame between the release of a job and its completion.

(Refer Slide Time: 21:00)



Selecting an Appropriate Frame Size (F)

- **Minimum scheduling overhead and chances of inconsistency:**
 - F should be larger than each task size.
 - **Sets a lower bound.**
- **Minimization of table size:**
 - F should squarely divide major cycle.
 - **Allows only a few discrete frame size.**
- **Satisfaction of task deadline:**
 - Between the arrival of a task and its deadline:
 - At least one full frame must exist.
 - **Sets an upper bound**


So, F should be larger than the tasks size otherwise it becomes complicated to run it because each time the scheduler wakes up on a periodic timer interrupt it must start execution of some job and F squarely divides the major cycle otherwise the schedule table will become large.

So, now, let us look at the third constraint between the arrival of a task and its deadline at least one full frame must exist. So, this sets an upper bound on the frame size.

(Refer Slide Time: 21:57)

Minimize Inconsistency

- Unless a job runs to completion:
 - Its partial results might be used by other jobs, leading to inconsistency
- To avoid scheduler overhead:
 - Selected frame size should be larger than execution time of each task.
 - Sets a lower bound for frame size.


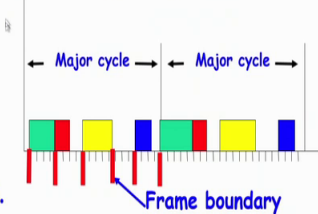


As we try to select a frame size we might find that multiple frame sizes are acceptable then we choose the largest frame size because that minimizes the overhead because the number of interrupts from the periodic timer become less number of time the scheduler runs. So, it is advantageous for us to select the largest frame size when multiple frame sizes are acceptable which meet the constraints.

(Refer Slide Time: 022:40)

Minimization of Table Size

- Unless the minor cycle squarely divides the major cycle:
 - Storing schedule for one major cycle would not be sufficient.
 - Schedules in the major cycle would not repeat:
 - This would make the size of the table large.



So, this point we had already said that unless the major cycle is squarely divided by the minor cycle and the frame then the schedule table length would become unnecessary

large, so in every design that we undertake ends here that the frame size squarely divides the major cycle or major cycle is integral multiple of the frame size.

(Refer Slide Time: 23:15)

Satisfaction of Task Deadline

- Between the arrival of a task and its deadline:
 - At least one full frame must exist.
- If there is not even a single frame:
 - The task would miss its deadline,
 - By the time it could be taken up for scheduling, the deadline could be imminent.

The diagram shows a horizontal timeline with two vertical lines marking the arrival time t_i and the deadline d_i . Below the timeline, two rectangular blocks labeled 'F' represent frames. The first frame is green and starts at t_i . The second frame is pink and starts at the end of the first frame. The deadline d_i is marked at the end of the second frame, indicating that the task's execution time extends beyond its deadline.

Now, let us see why there must be at least one full frame between the arrival of a task instance that is a job and its deadline. Let us take this instance.

Let us say we have this task instance I arrives here just after the frame starts. Obviously, this frame cannot start execution of this task because the scheduler activates only at the start of the frame and if anything if execution is to be started must be undertaken at this point. So, the execution of the t_i cannot be started in this frame. A scheduler can only take up its execution the next frame, but then by that time its deadline is very near. So, the task time may extend beyond the deadline. So, there is a chance that the deadline will get missed and that is the reason why in our design we will insist that the frame size must be set such that there is a clear frame between the release of a task and its deadline and that sets the largest size of the frame.

(Refer Slide Time: 24:55)

Satisfaction of Task Deadline

- The worst case for a task occurs when the task arrives just after a frame has started.

The diagrams show a horizontal timeline. In the top diagram, a green bar represents a frame starting at time 0 and ending at time F . A pink bar represents a task instance starting at time t_i and ending at time d_i . The task instance starts after the frame has begun, and the text indicates it would miss its deadline. In the bottom diagram, the task instance starts at time 0, before the frame begins, and the text indicates its deadline can be met.

69

So, now our design problem is that how to decide which is the minimum size of the frame sorry which is the maximum size of the frame that we must choose based on this constraint that there must be a clear frame which must exist between the release of task instance and its deadline. Obviously, if our frame size is larger than the task instance and its deadline we may not have clear frame which exists between these but that can we do better than that.

(Refer Slide Time: 25:40)

Satisfaction of Task Deadline

- The minimum separation of arrival time of t_i from a frame start:
 - $\text{GCD}(F, p_i)$
- Thus, for all t_i
 - $2F - \text{GCD}(F, p_i) \leq d_i$ must be satisfied

The diagram shows a horizontal timeline. A pink bar represents a task instance starting at time t_i and ending at time d_i . Two green bars represent frames of size F , one starting at time t_i and another starting at time d_i .

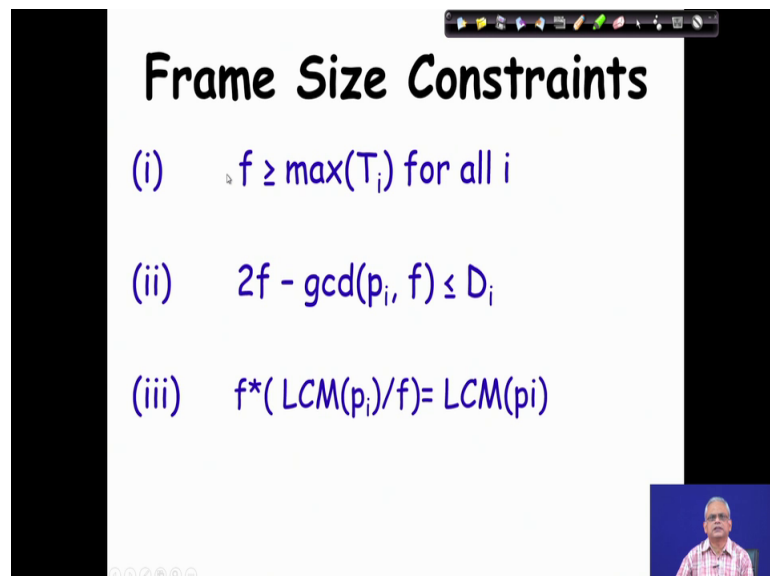
70

We will come up with simple expression that $\text{GCD}(F, p_i)$ will not go into the details of the derivation of this expression $\text{GCD}(F, p_i)$ gives the size of the frame. So, $2F - \text{GCD}(F, p_i) < d_i$ must be satisfied. So, now, worst case occurs for any task given by the $\text{GCD}(F, p_i)$. So, it may at most occur after $\text{GCD}(F, p_i)$ of a frame this is the worst case. And in this case for the task to meet its deadline we must have its deadline before the end of the second frame because we may at most start its execution here.

So, this much time has elapsed and this is given by $F - \text{GCD}(F, p_i)$ and we have just one more frame is remaining. So, $2F - \text{GCD}(F, p_i) < d_i$ must be satisfied for every task because $\text{GCD}(F, p_i)$ indicates the worst case scenario where a task occurs just after the frame has passed and therefore, the remaining time here for which the task cannot be started execution given by $F - \text{GCD}(F, p_i)$ and then the task may be taken up here and have just 1 F left.

So, the deadline if it here at the end then we will be adjustable to meet it and that is what is expressed here $2F - \text{GCD}(F, p_i) \leq d_i$ that must be for every task.

(Refer Slide Time: 27:56)



Frame Size Constraints

- (i) $f \geq \max(T_i)$ for all i
- (ii) $2f - \text{gcd}(p_i, f) \leq D_i$
- (iii) $f * (\text{LCM}(p_i)/f) = \text{LCM}(p_i)$

So, these are the three constraints. The frame size must be more than every task execution time, $2F - \text{GCD}(p_i, F) < d_i$ for every task and the LCM of all periods which is the major cycle must squarely divide the must be squarely

divided by the frame size that we write in this expression, F into $LCM \pi$ divided by F must be equal to $LCM \pi$. So, this indicates that the frame size squarely divides the major cycle. So, these are the three constraints. So, you must ensure when we want to select the frame size.

(Refer Slide Time: 28:48)

Selection of a Suitable Frame Size

- Several frame sizes may satisfy the constraints:
 - **Plausible frames.**
- **A plausible frame size has been found:**
 - **Does not mean that the task set is schedulable.**
- The largest plausible frame size needs to be chosen:

Will find that several frame sizes are becomes possible. So, this we call as plausible frame sizes and then even if we have a plausible frame size it is not the case that schedule may be found may be because we have the total number of job instances to be handled is more than the number of frames in a major cycle. So, we have to be careful here all plausible frame sizes may not be acceptable will see through some examples and then we find the frame sizes which can be used and then we choose the largest of those because that will minimize the overhead due to the scheduler.

So, we will stop here we will start with few examples of designing the frame size and assignment of tasks to the frames in the next lecture, now we will stop.

Thank you.