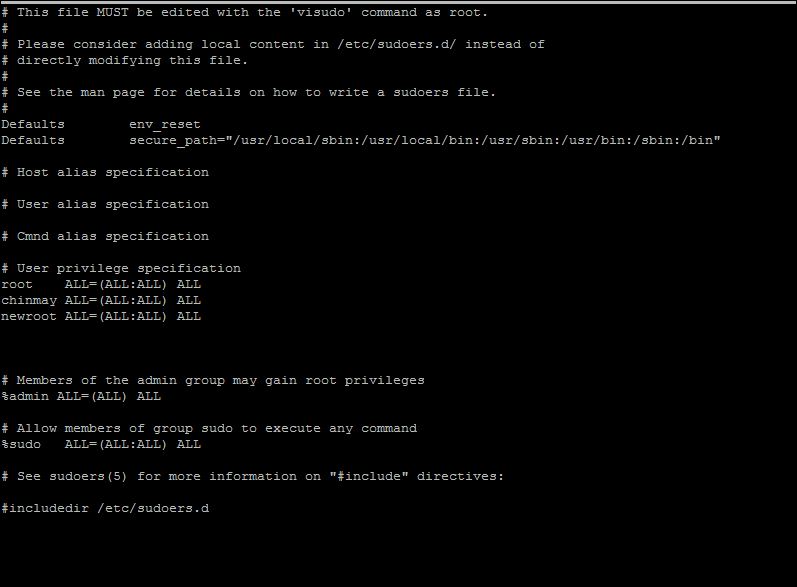
Lab Exercise 2 –– Service Scheduling Feasibility

Observations and Analysis

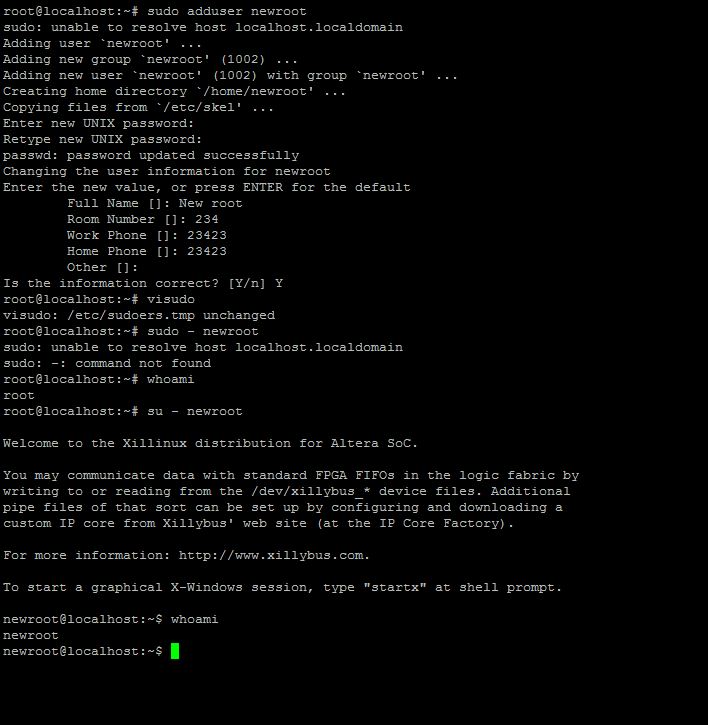
1.

**Ans**

I have created two new users newroot and chinmay. This new user is added using “sudo addsuer” (FigB) and giving permissions as root by using visudo(FigA). The screenshot of command “whoami” user FigA.



*Fig A Addition of sudo privileges using visudo*



*Fig B Screen shot of whoami command*

2.

**Ans**

**Summary**

The paper states [4], [3] that NASA Space Shuttle Flight software uses frequency (cyclic) executive design for PASS (Primary Avionics Subsystem) and has been providing a defect free hard real time solution for a very complex system.

Cyclic executive does not require a RTOS instead it is a loop control structure to schedule multiple (execute more than one) periodic process on a single CPU [3]. As in PASS software includes different hard real time safety services such as GN&C (guidance, Navigation and Control), SM (Systems Management), VCO (Vehicle Checkout) and others. The GN&C cyclic executive is scheduled to run periodically at 25 Hz which having its own dispatch table to sequence its own higher frequency (VCO) at relatively higher priority and mid and lower (SM) frequency at lower priority [4].

According to Carlow [3], the flight software is a synchronous approach allowing a dispatcher to always schedule jobs at same time (point of time) relative to start of overall system

However, this efficient frequency executive as mentioned in [3], [4] and [5] have following advantages and disadvantages.

**Advantages over other scheduling executive**

1. **Predetermined (Less possibility of missing Deadlines)**
   1. The entire execution of schedule is predetermined such as it meets all its response time requirements. Thus fulfills basic need of hard real time systems of meeting its deadlines
   2. This comes with an assumption when cyclic /frequency executive does not have any overrun of frames due to external /other events
2. **Low Context Switching (No Preemption requirement)**
   1. As each task in this scheduling a fixed amount of slice of time is allotted to each task, there is no requirement of priorities and preemption.
   2. Thus are no context switches and interrupts which are unexpected keeping low overhead time of execution.
   3. Further it reduces need protection of shared resources using mutex and semaphores leading to a simple and maintainable software.
3. **Low Jitter**
   1. Determinism nature of schedule throughout the period of execution of all tasks leads to low jitter
   2. As cycle to cycle deterministic period of execution for each task is performed, the jitter is sustained to be low (variation in time a computed result is output to external environment [5]).
   3. This is also due to non-interference of interrupts and non- preemption of tasks (point 2).

**Dis-Advantage over other scheduling executive**

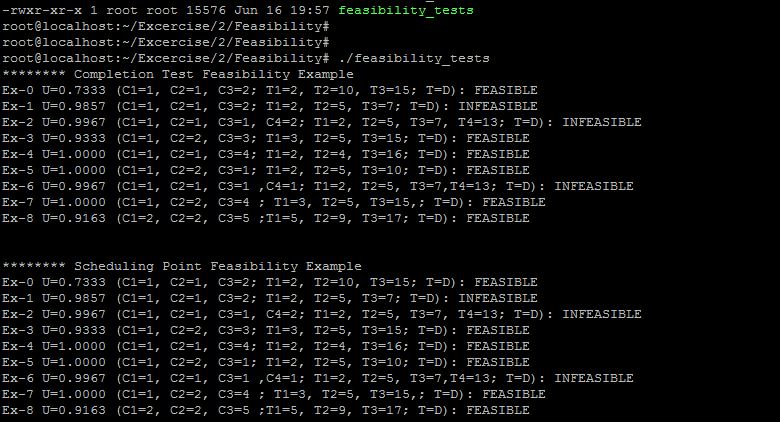
1. **Less Flexibility (Difficult to modify)**
   1. It provides less flexibility to code /application ([4], [3]) and a fragile application [5]
   2. As cyclic/frequency executive assigned after careful consideration requirement of each task. When an another sensor is added(task), the complete design of schedulable task has to be changed, thus many careful changes and investment of time in redesign and testing.
   3. If not many changes are made to initial design, the application is fragile. As some part of the system will overrun its frame time bound during maintenance (or cause software errors) for incorporating a new task (eg. sensor).
2. **Preemption for Handling of functions whose execution timer is long compared to the period of the highest rate cyclic task [5]**
   1. A common situation occurs when a sensor with slow sampling rate requires higher computation time and higher sampling rate task requires lower computation time
   2. A solution this problem is dividing longer computational task into smaller tasks.
   3. It in turn introduces preemption and overhead leading to interferences as other scheduling techniques.
3. **Harmonic frequency task requirement** 
   1. The limitation (a hidden cost [5]) for using frequency executive approach is tasks have to share a harmonic relationship.
   2. If the harmonic tasks are not scheduled it leads to underutilization of CPU

The disadvantages overpower advantages and thus we look into other scheduling techniques for a simpler and tighter solution

3.

Ans.

After running the code for RMS using scheduling point and completion time to check feasibility of code. The Ex1, Ex2, Ex6 are examples are infeasible using RMS scheduling (Fig C).



***Fig C****- Screen shot of Feasibility code for Example (Ex0 – Ex8) –RMS using Scheduling and Completion Test*

*\*\* Code and cheddar projects have been also submitted separately*

Task set of Ex-1, Ex-2 and Ex-6 are not schedulable (not feasible) using RMS whereas are schedulable using EDF and LLF as seen from cheddar analysis. As can been as marked (Task 4) Failure in Fig D (RMS). The Question arises Task 3 does not have urgency to be executed at this point of time. As we have observed that RMS schedules priorities of tasks only according to frequency of occurrences (T) of tasks.

While EDF (Earliest Deadline first) schedules or changes priority **dynamically** according to deadline (amount of time left for executing the job) as seen from **Fig D** ((EDF)and timing diagram for Ex2) and applied in Ex1, Ex2.

Further using LLF (Least Laxity First scheduling algorithm) which defined as Laxity = Deadline – Remaining computation /Execution time. In this scheduling algorithm task having lower value Laxity is assigned higher priority (As seen from Fig D(LLF)).

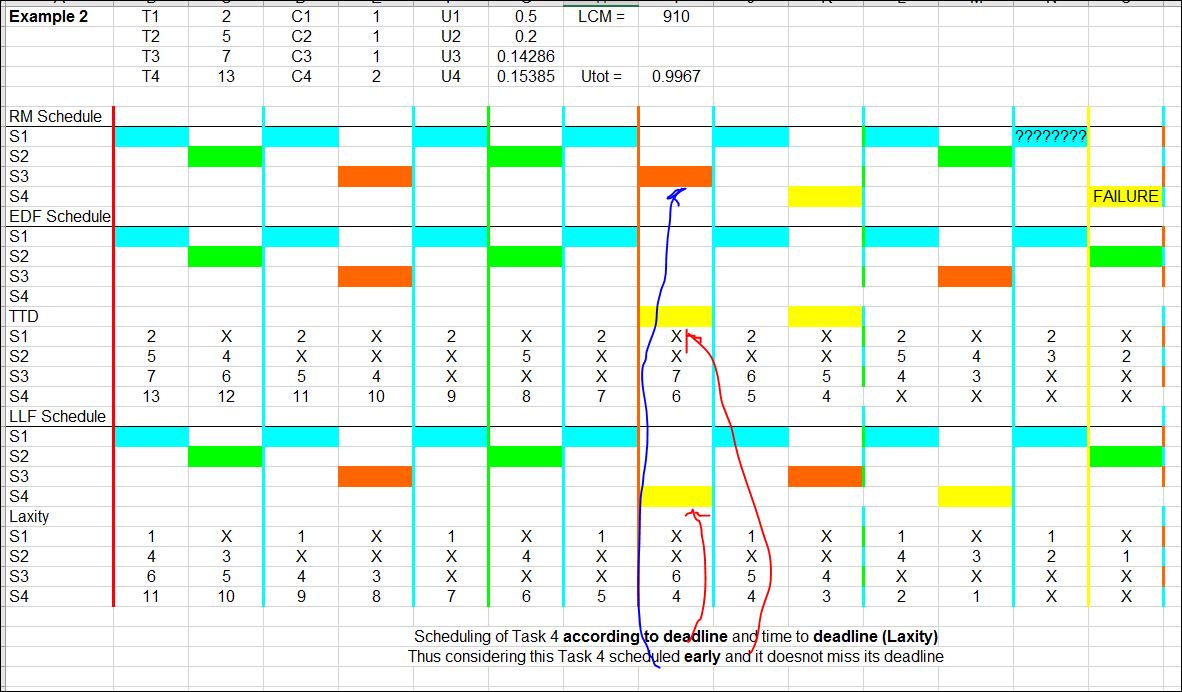


Fig D) Depicting advantage of EDF and LLF where RMS failed

**Cheddar Analysis**

(i)

The Examples were executed on cheddar tool (Cheddar 2.0 as LLF algorithm working incorrectly Cheddar 3.0, as suggested by TA Akshay Singh and Sairam) and its screen shots are below. Screen shots of RMS, LLF and EDF scheduling algorithm for Example 0 to 8 are as below.

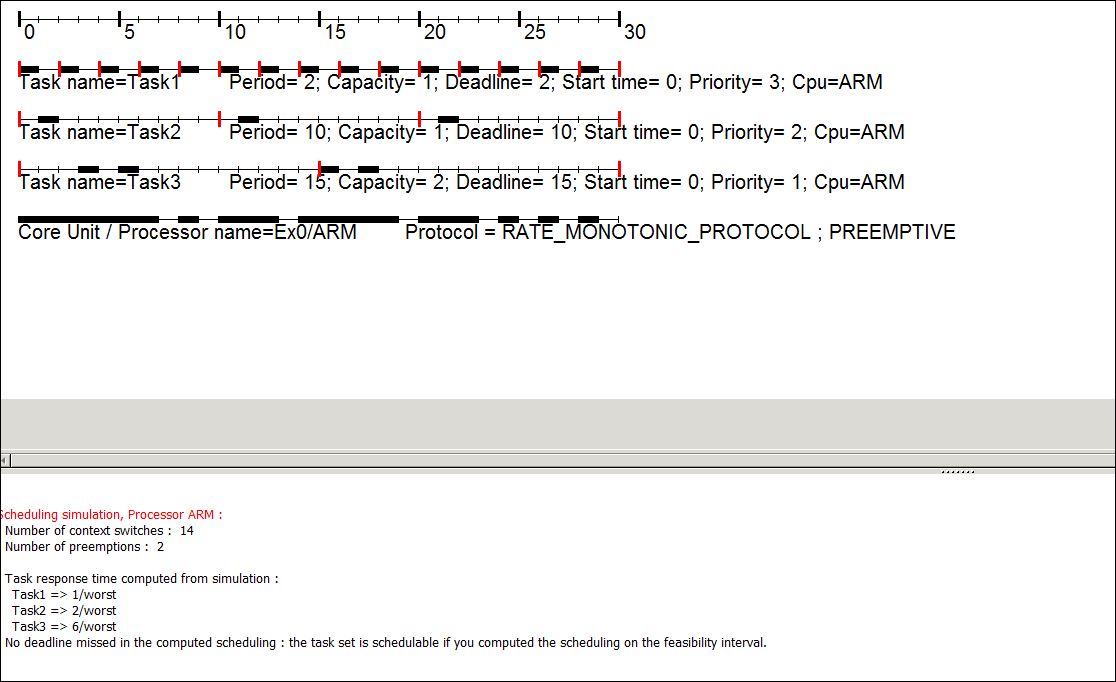
(ii)

Cheddar analysis agrees with Code(RMS) and Hand drawn examples (LLF and EDF) as this tool uses all service simulations over the LCM and completion and scheduling tests within the tool to test all the scenarios including the calculation of utilization factor to be maximum 1.

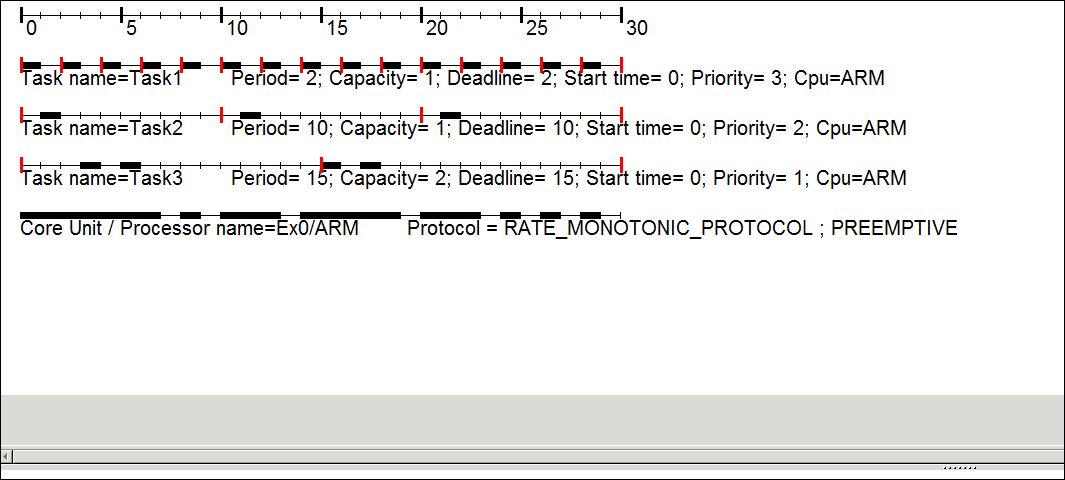
(iii)

Interesting results are obtained when deadlines are not same as period (Ex6) for EDF and LLF.Cheddar tool is also considering Utilization w.r.t deadline.

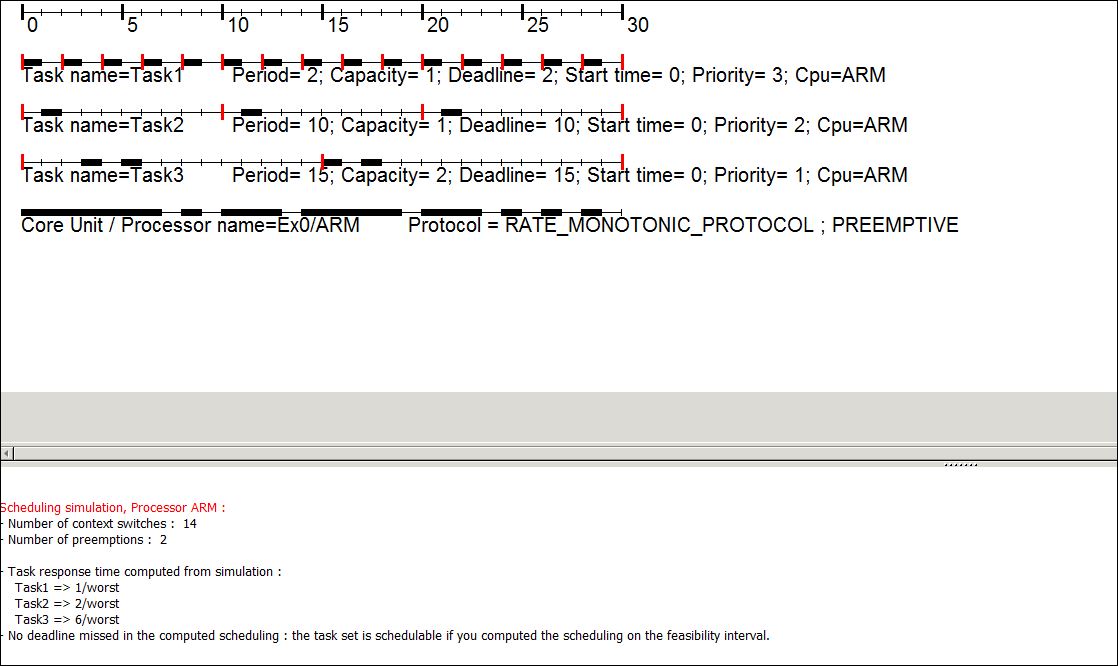
**Example0:**



*Fig1.1* ***Feasible****(schedulable) Example 0 using RMS*



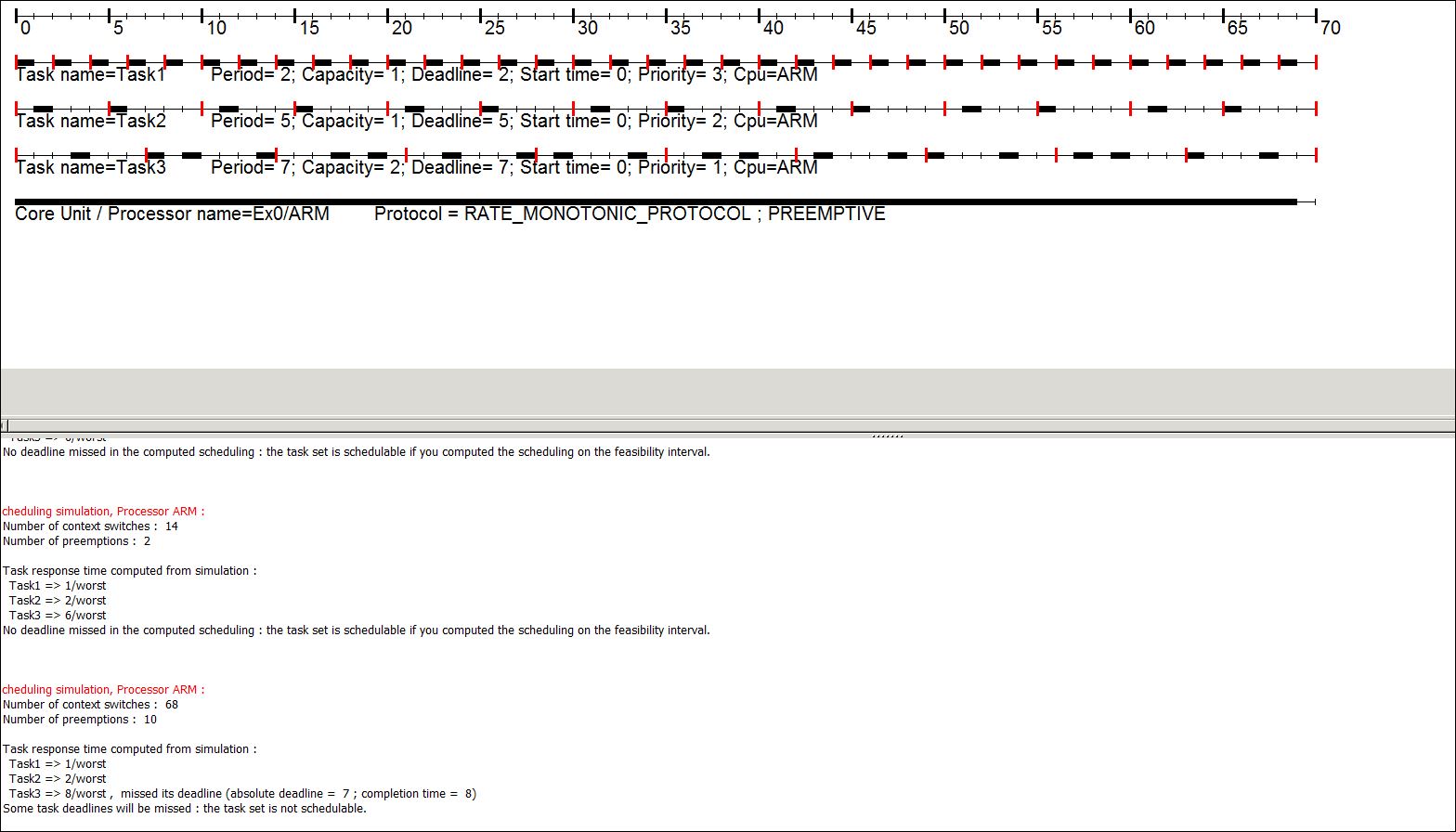
*Fig1.2* ***Feasible****(schedulable) Example 0 using EDF*



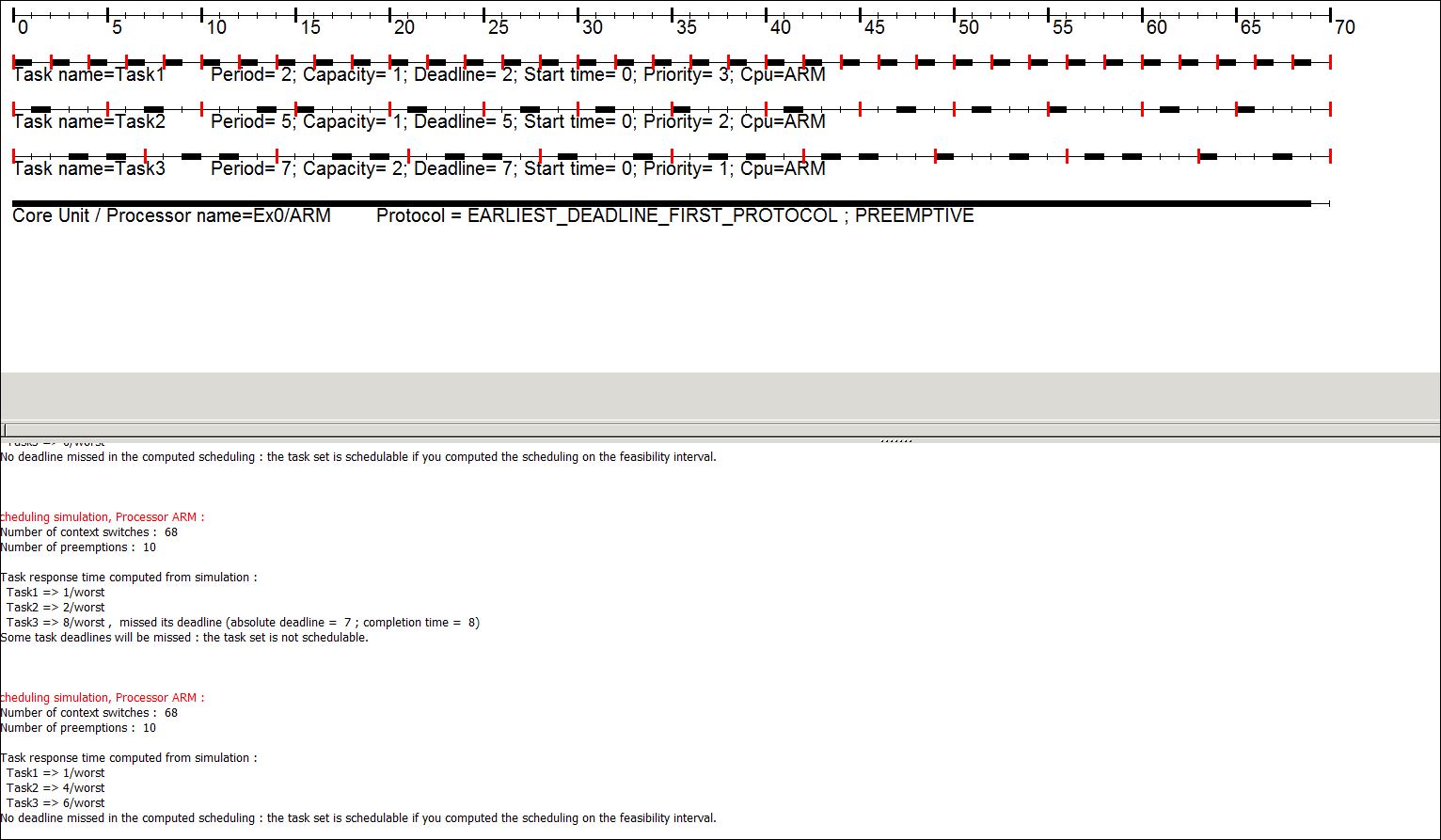
*Fig1.3* ***Feasible****(schedulable) Example 0 using LLF*

**Example 1:**

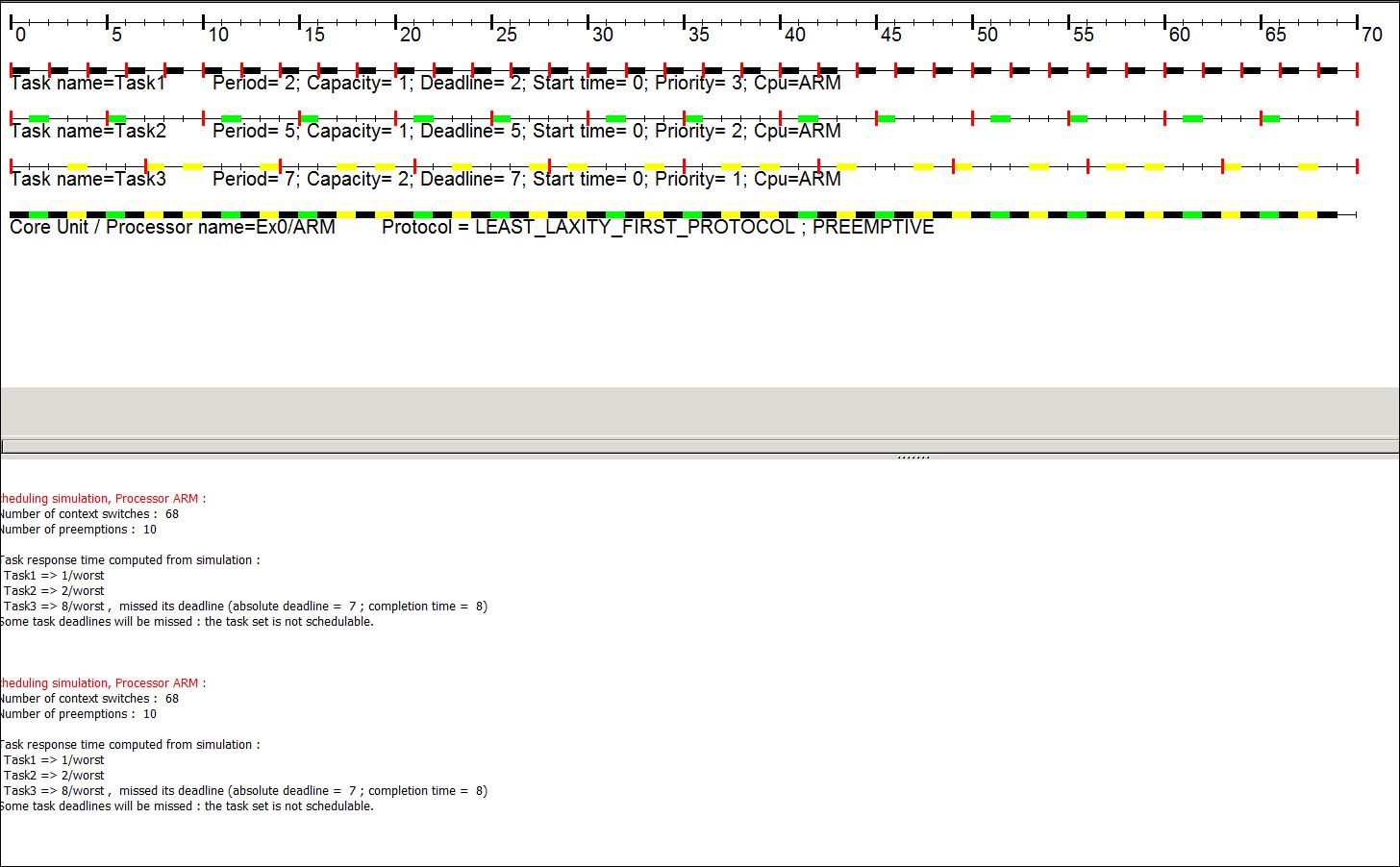
This Task Set is **not schedulable** using RMS whereas schedulable using EDF and LLF (Fig1.4,1.5) using Cheddar analysis.



*Fig1.4* ***Not Feasible*** *(not schedulable) Example 1 using RMS*



*Fig1.5* ***Feasible****(schedulable) Example 1 using EDF*



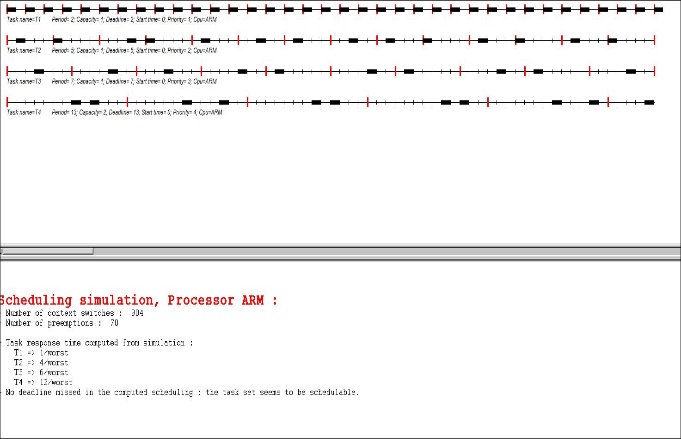
*Fig1.6* ***Feasible****(schedulable) Example 1 using LLF*

**Example 2:**

(i)This sample feasibility is checked for 910(LCM) units, however only plotted for 70 units of time. It is **not schedulable** using RMS (Fig2.1) whereas schedulable using EDF and LLF (Fig2.2, Fig2.3) using Cheddar analysis.

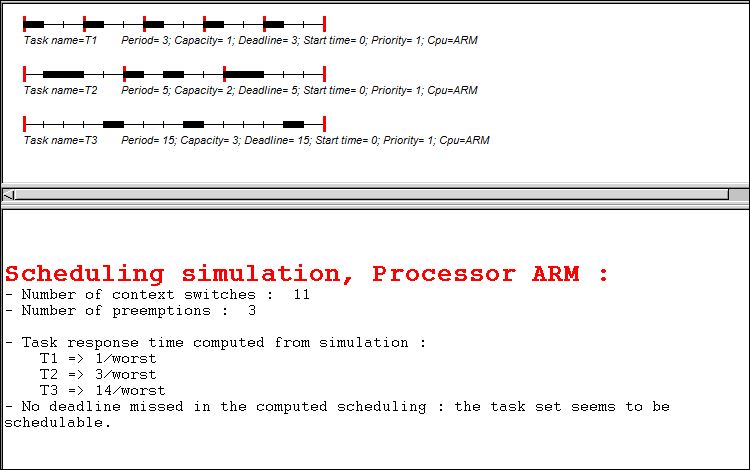
****

*Fig 2.1* ***Not Feasible****(schedulable) Example 2 using RMS as Deadline is missed*

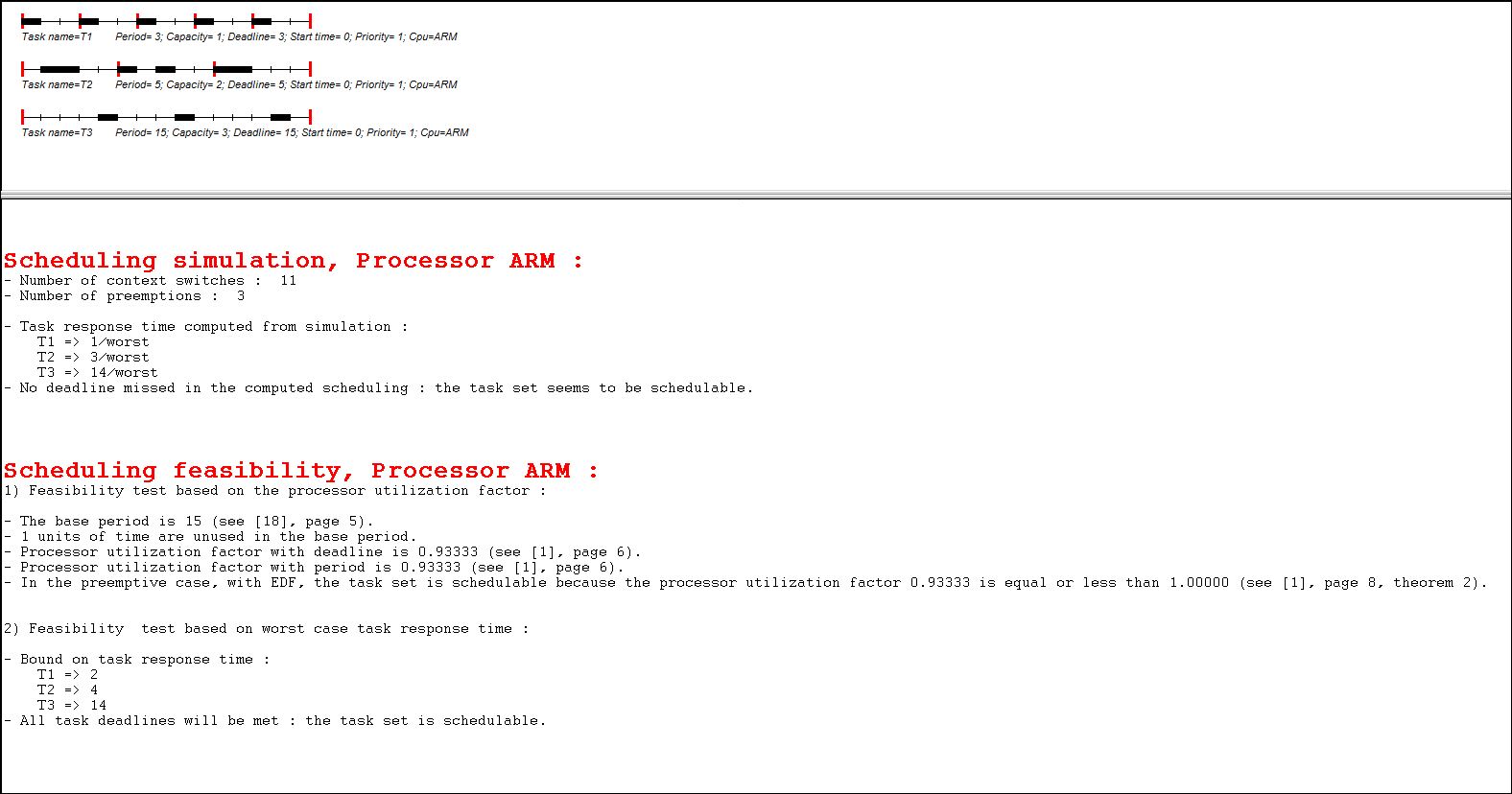
* *

*Fig 2.2 Feasible(Schedulable) Example 2 using EDF Fig 2.3 Feasible(Schedulable) Example 2 using LLF*

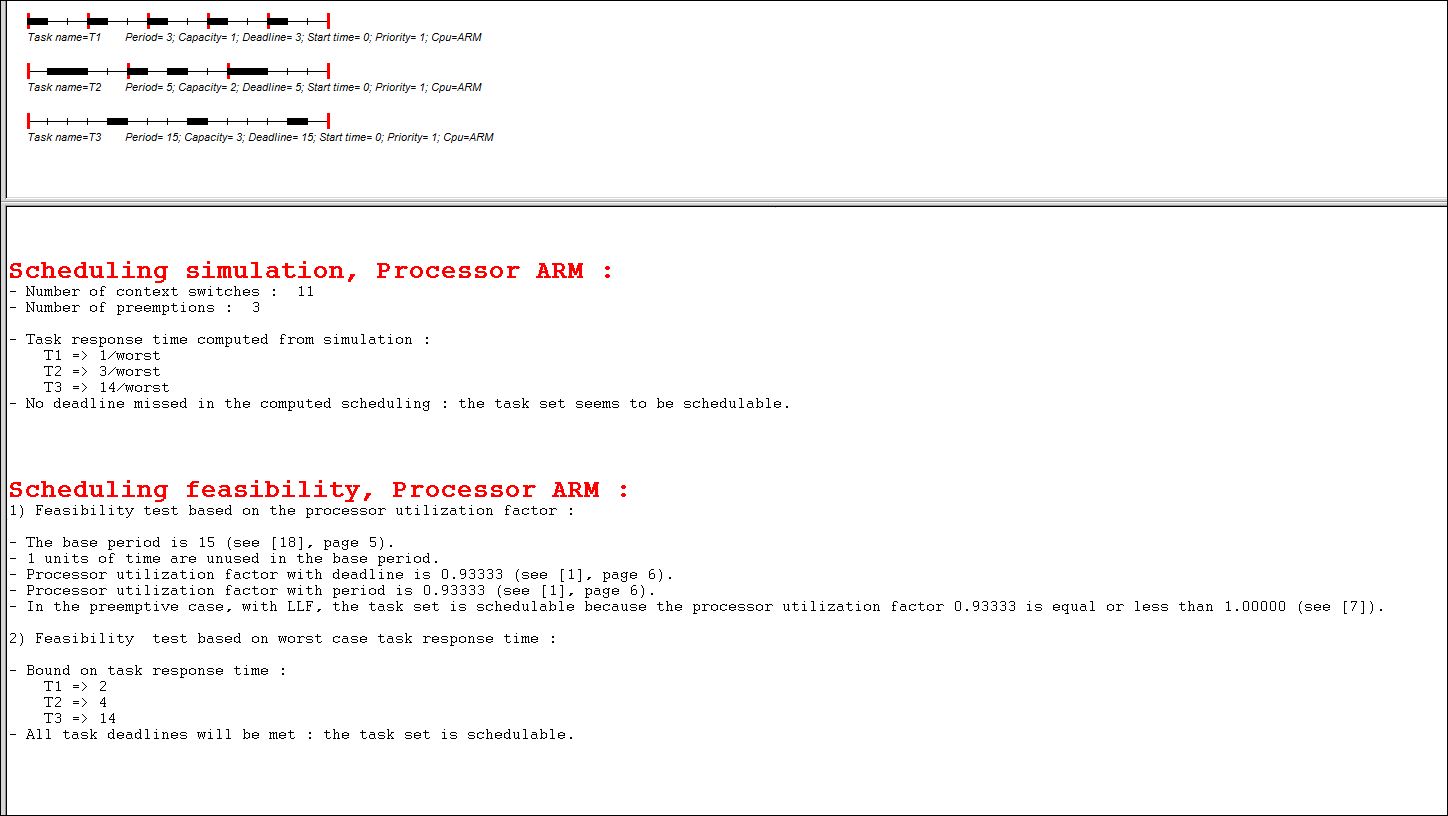
**Example3:** This set of tasks are schedulable according to cheddar using all policies



*Fig 3.1* ***Feasible****(schedulable) Example 3 using RMS*

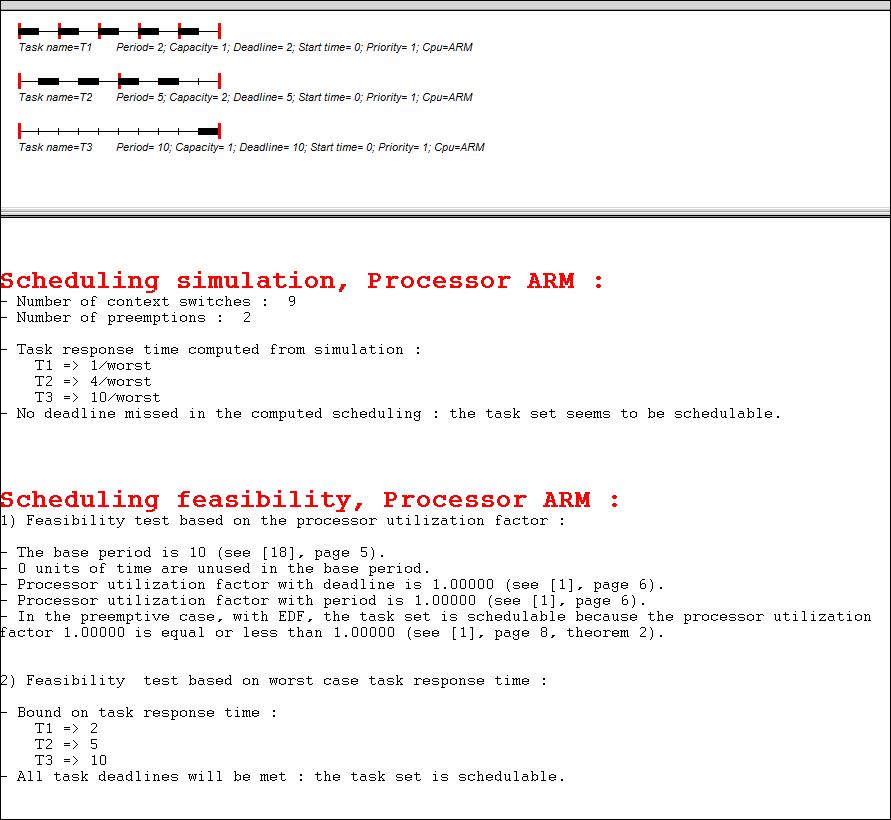
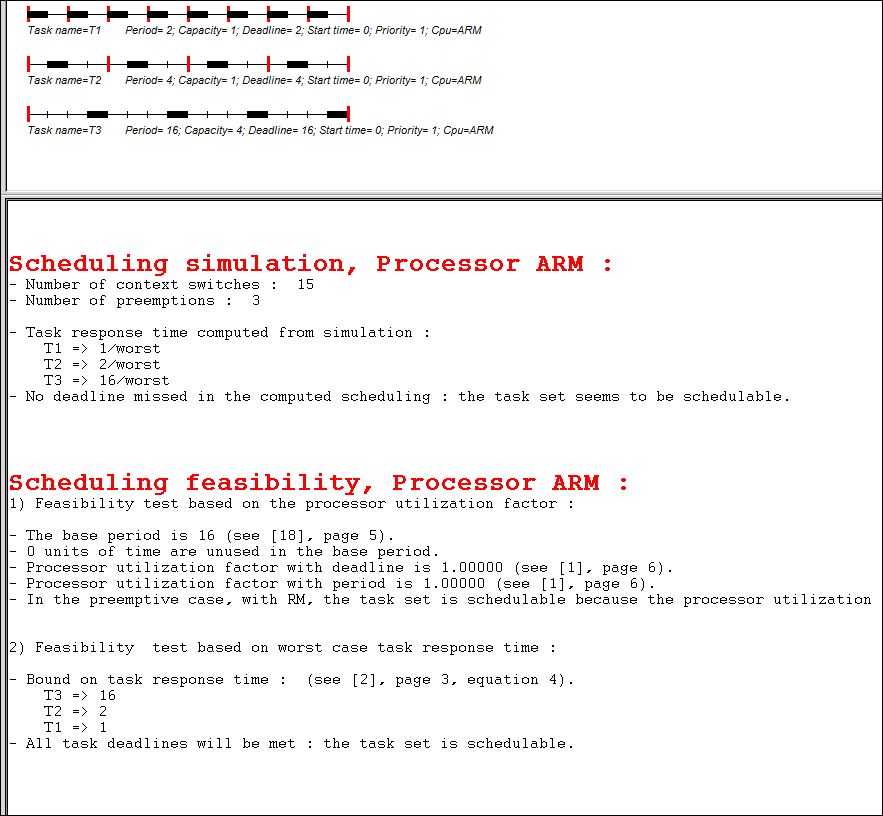


*Fig 3.2* ***Feasible****(schedulable) Example 3 using EDF*

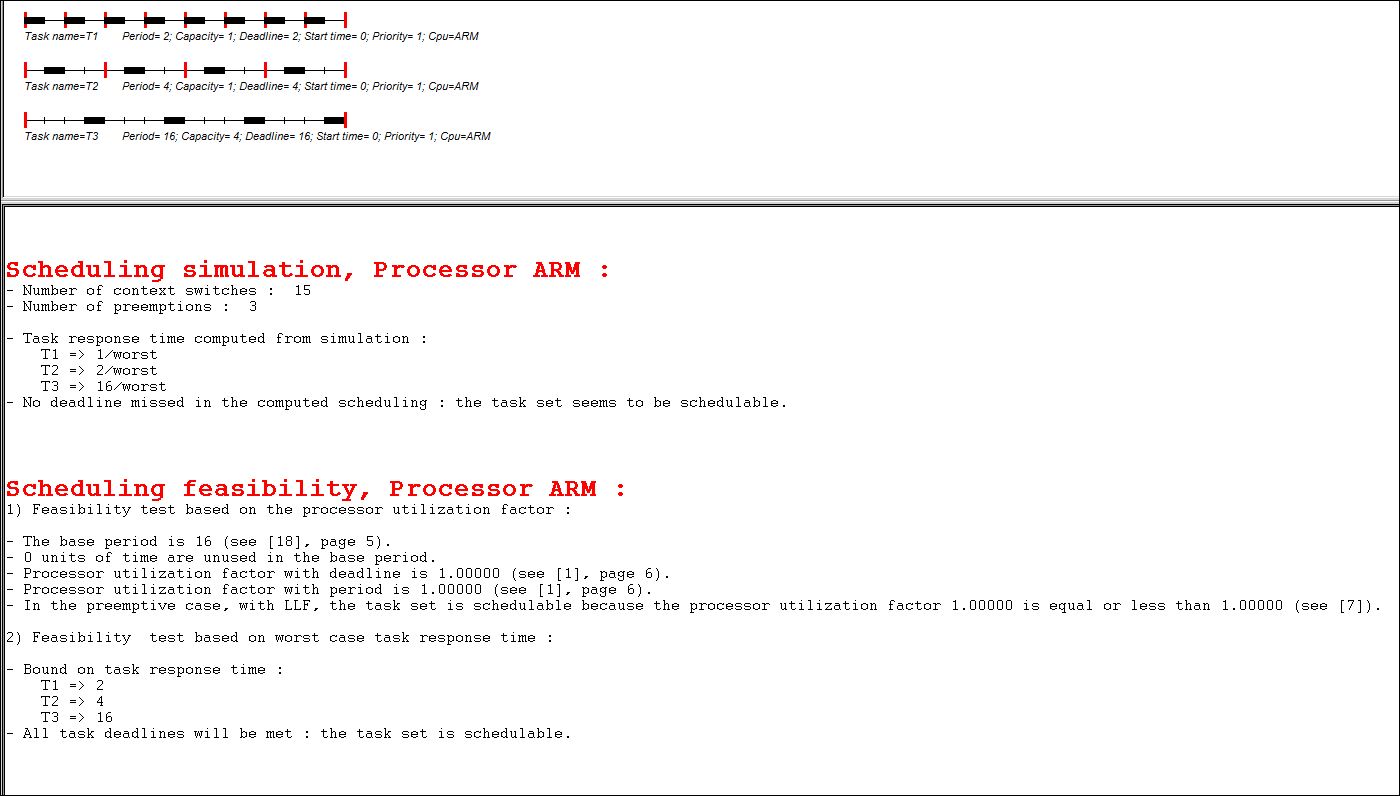


*Fig 3.3* ***Feasible****(schedulable) Example 3 using LLF*

**Example4:** This set of tasks are schedulable according to cheddar using all policies

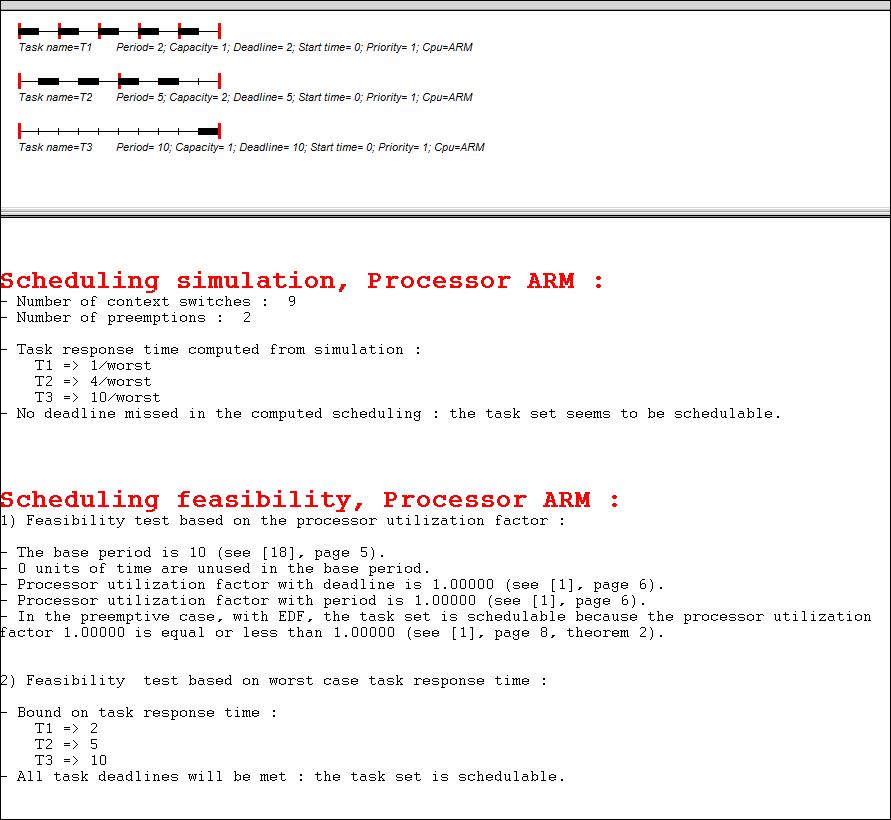
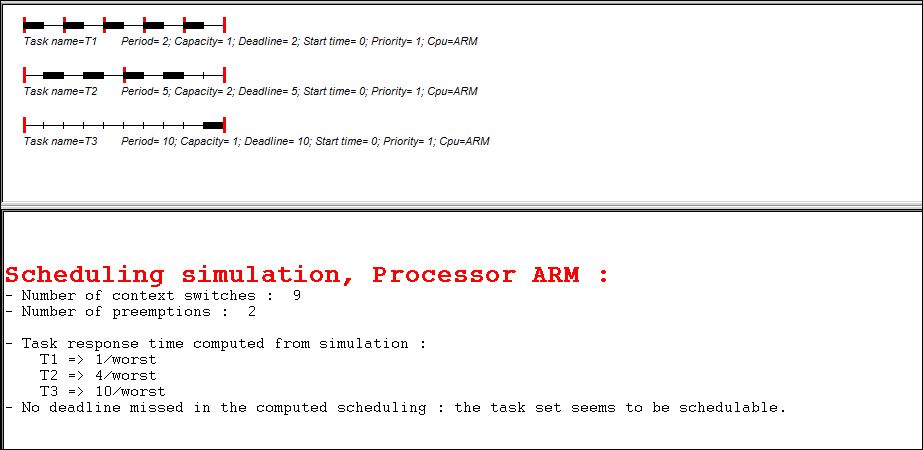


*Fig 4.1 Feasible Example 4 using RMS Fig 4.2 Feasible Example 4 using EDF*

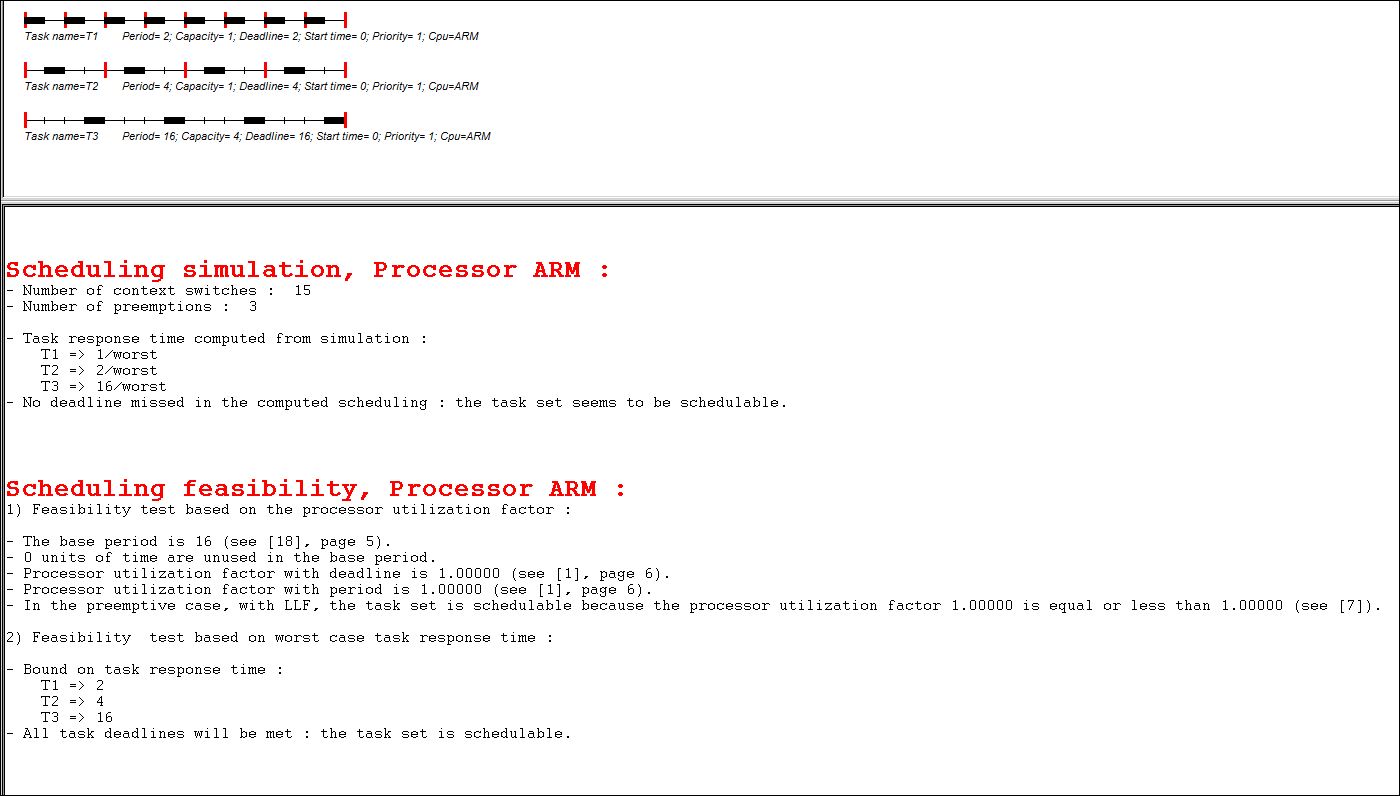


*Fig 4.3 Feasible Example 4 using LLF*

**Example5:** This set of tasks are schedulable according to cheddar using all policies



*Fig 5.1(i) Feasible Example 5 using RMS Fig 5.2(ii) Feasible Example 5 using EDF*



*Fig 5.3(ii) Feasible Example 5 using LLF*

**Example 6:**

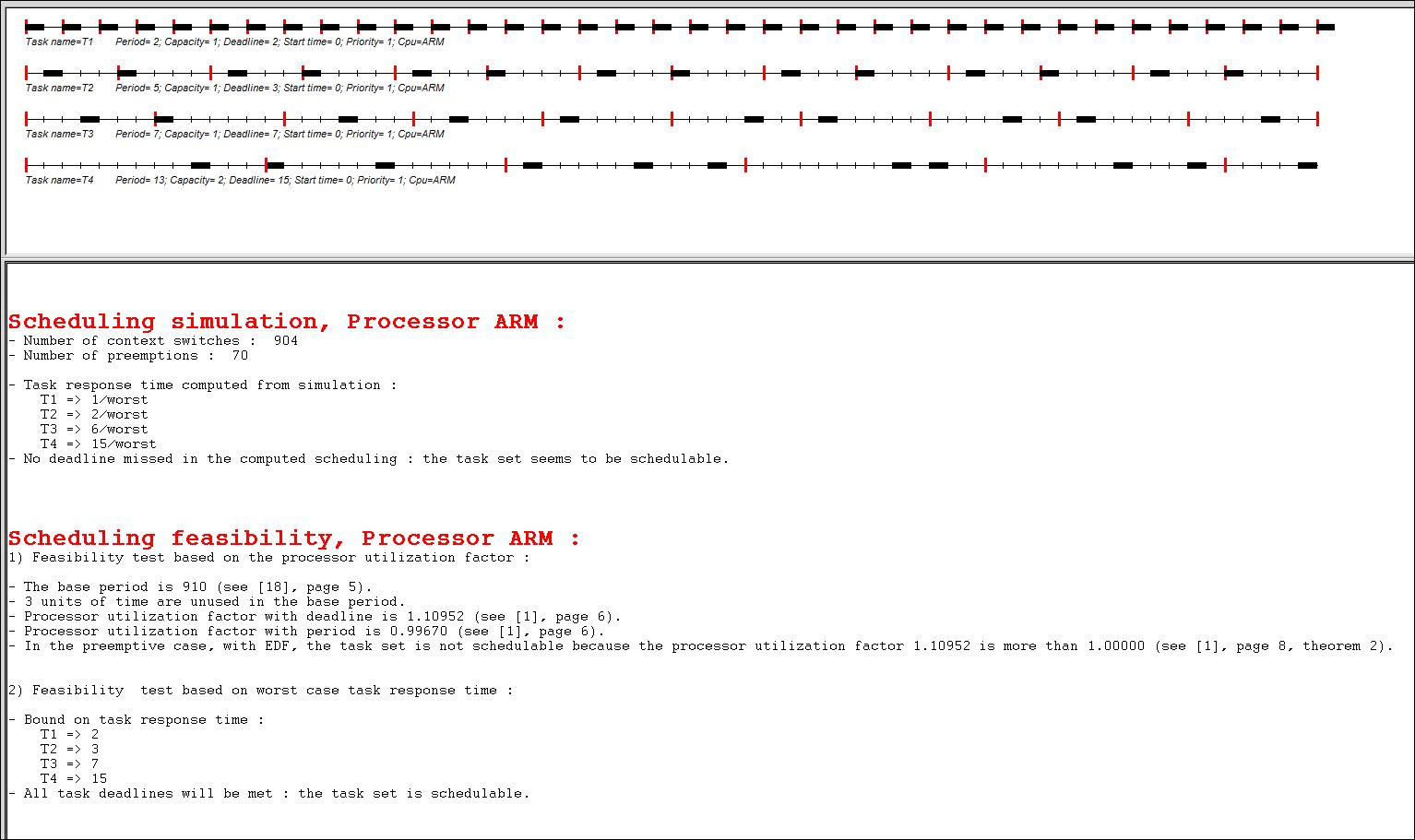
1. This sample feasibility is checked for 910(LCM) units, however only plotted for 70 units of time. It is **not schedulable** using RMS(Fig6.1) whereas schedulable using EDF (Ex 6.2) and not schedulable for preemptive case using LLF (Ex 6.3) using Cheddar analysis.
2. However, when utilization of EDF and LLF w.r.t to deadline are considered, U is greater than 1, which is violating and overloading CPU. Thus it is not feasible using EDF and LLF.
3. The Deadline used same as time period for RMS such as D1=T1=2, D2=T2=5, D3=T3=7 and D4=T4=13.
4. For EDF and LLF other deadlines have been incorporated as mentioned

D1=2, D2=3, D3=7, D4=15.

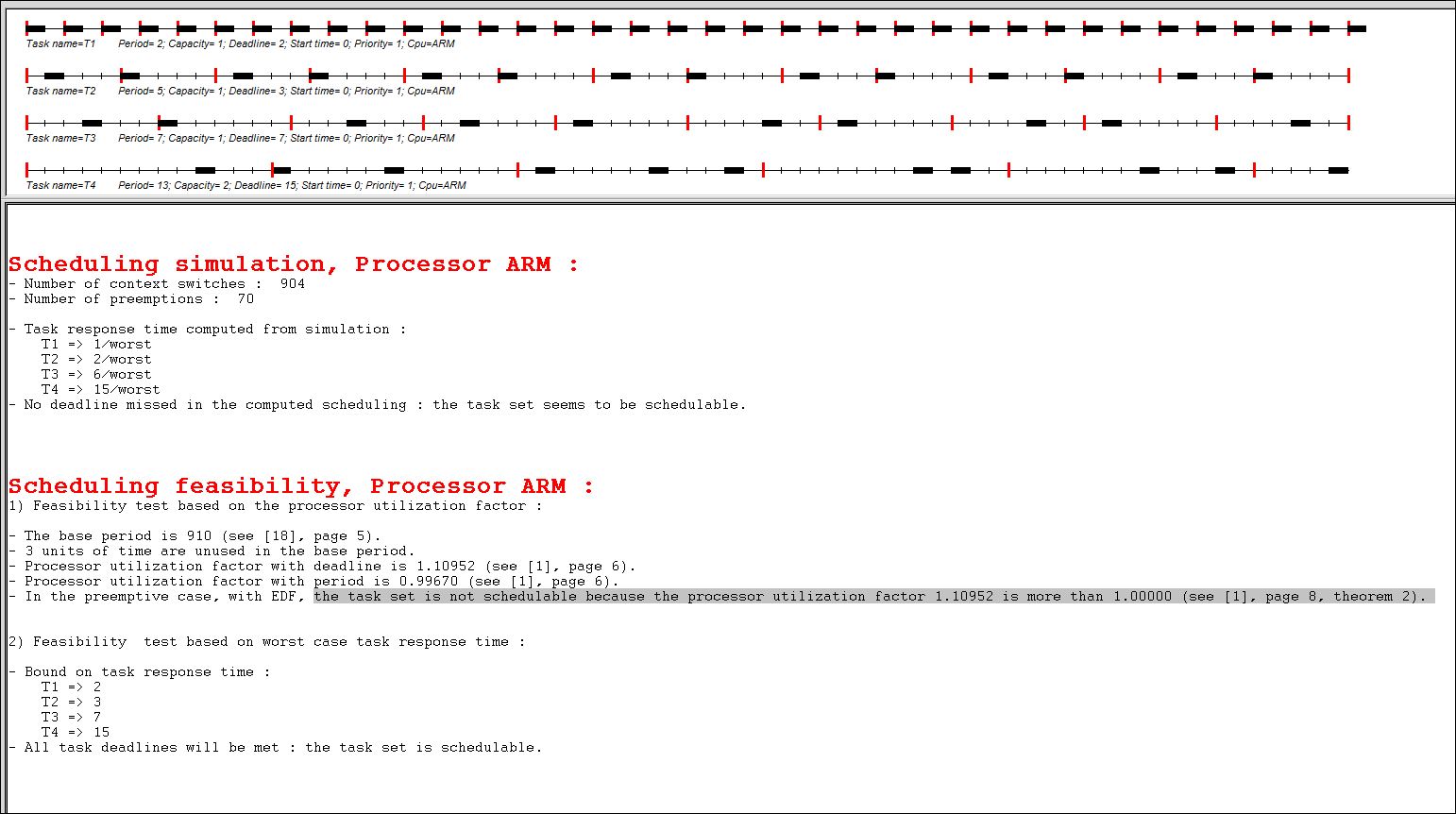
1. Hand drawn can be seen in Fig 6.4



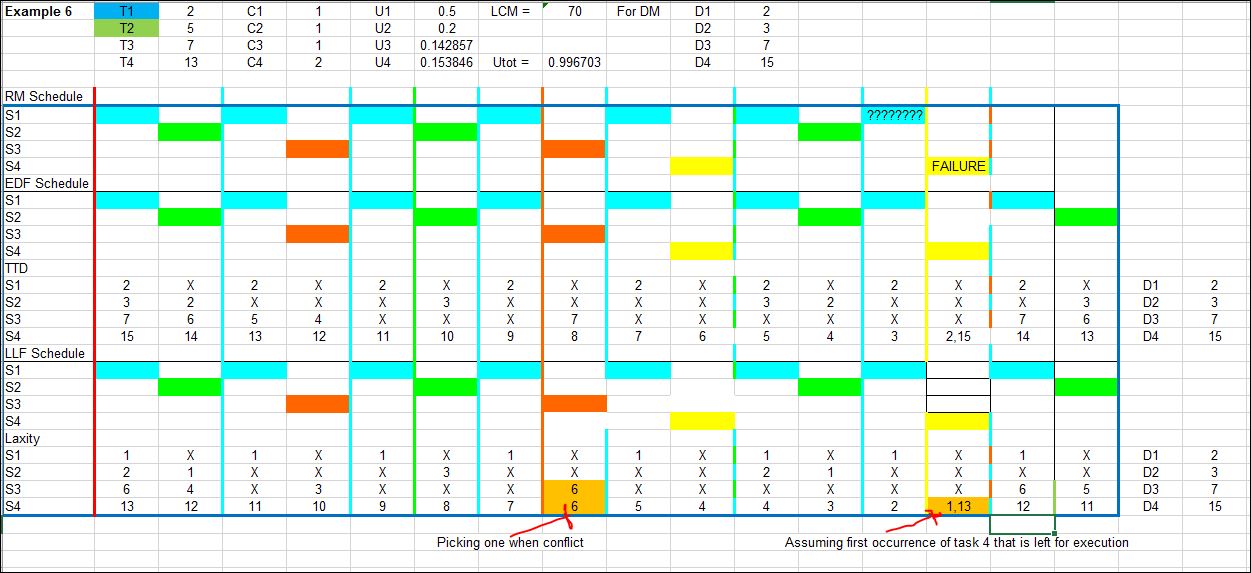
*Fig 6.1* ***Not Feasible*** *(not schedulable) Example 6 using RMS as Deadline is missed*



*Fig 6.2* ***Feasible(****schedulable) Example 6 using EDF (D1=2, D2=3, D3=7, D4=15)*



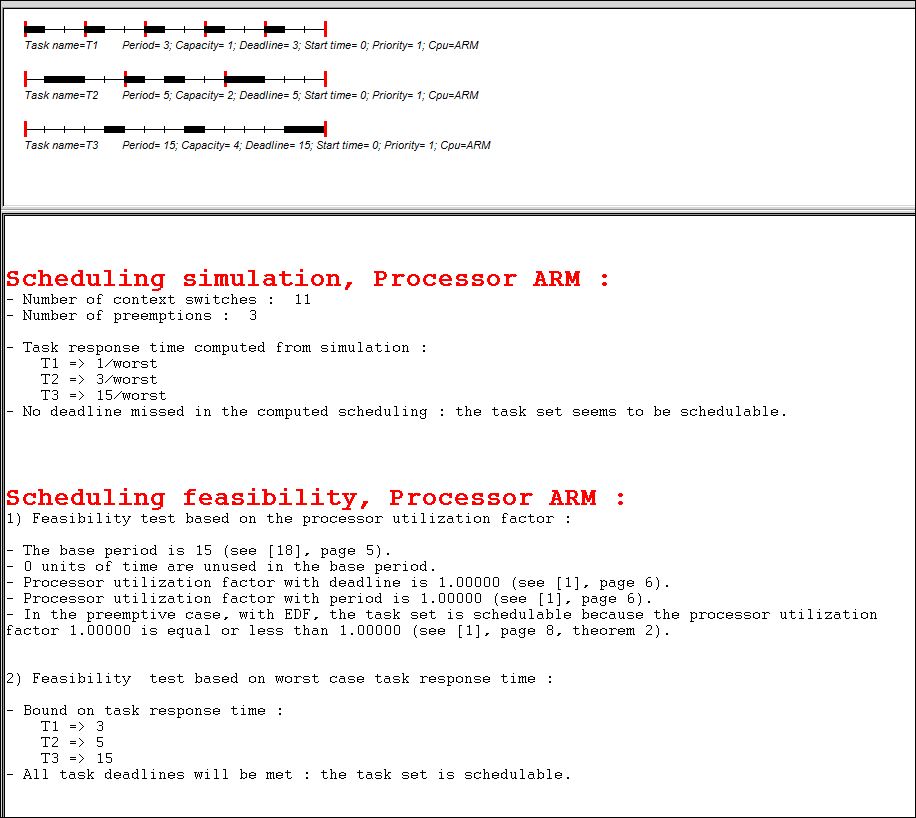
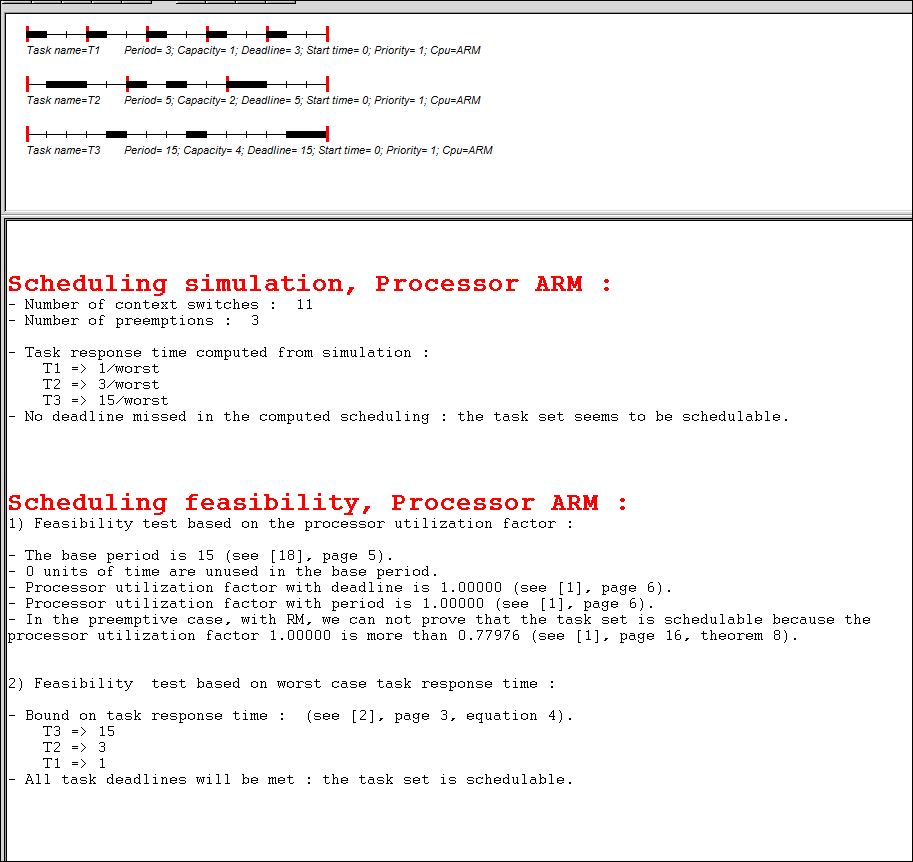
*Fig 6.3* ***Feasible(****schedulable) Example 6 using LLF (D1=2, D2=3, D3=7, D4=15) and* ***Not Feasible(schedulable)*** *for preemptive case as processor utilization (1.10952 > 1.00)*



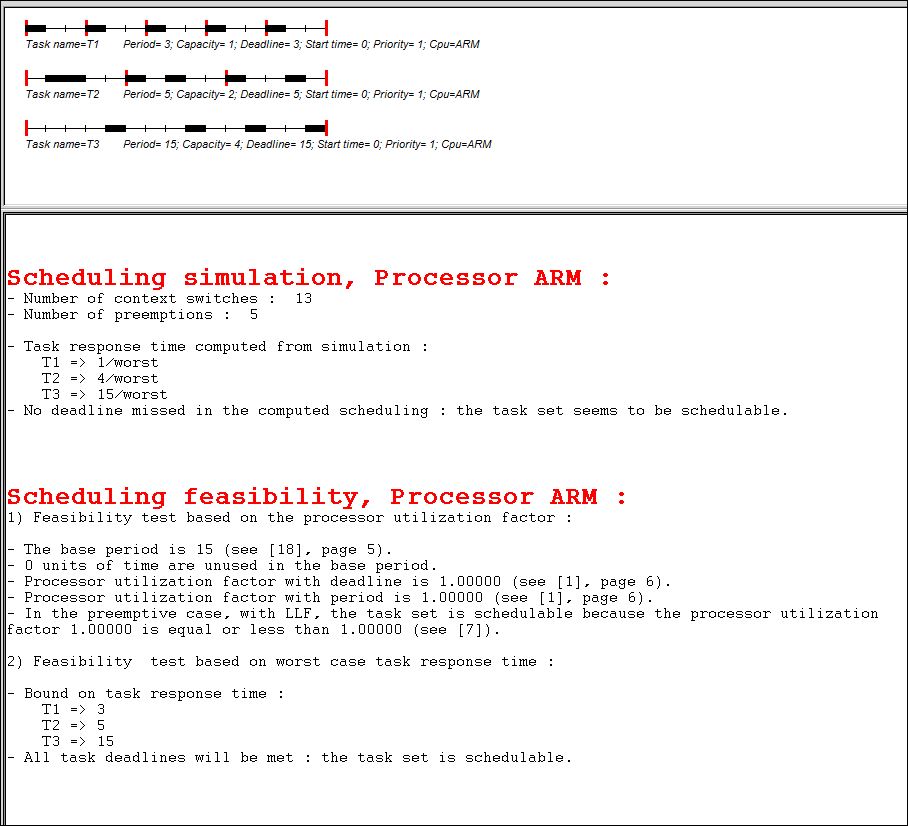
*Fig 6.4 Timing Diagram of example 6 Set Task*

(v)As can be seen from Fig 6.4 there are two interesting conflicts marked where same laxity for different task are assigned using LLF and two occurrences of same task are available with different laxity. For former scenario randomly task3 has been given higher priority and latter scenario first occurrence is considered

**Example 7:** This set of tasks are schedulable according to cheddar using all policies



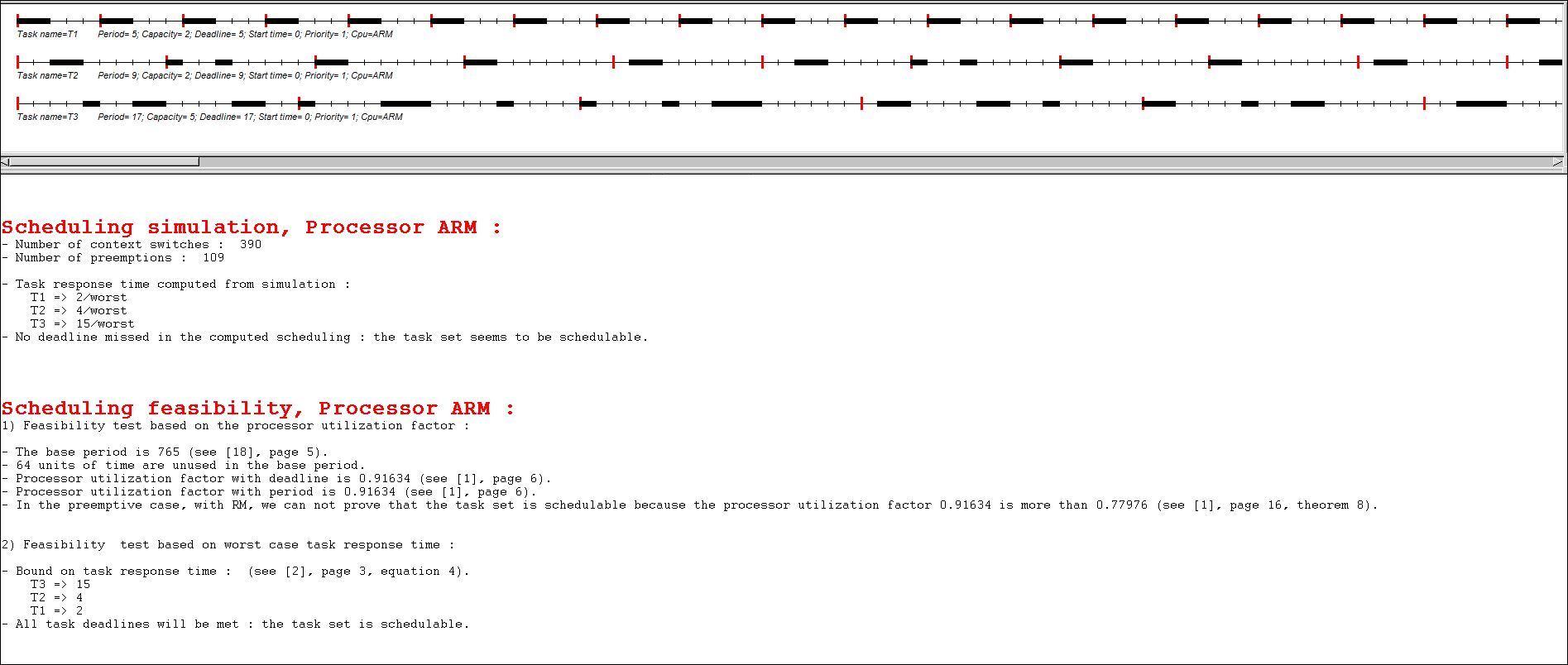
*Fig 7.1* ***Feasible****(schedulable) Example 7 using RMS Fig 7.2* ***Feasible****(schedulable) Example 7 using EDF*



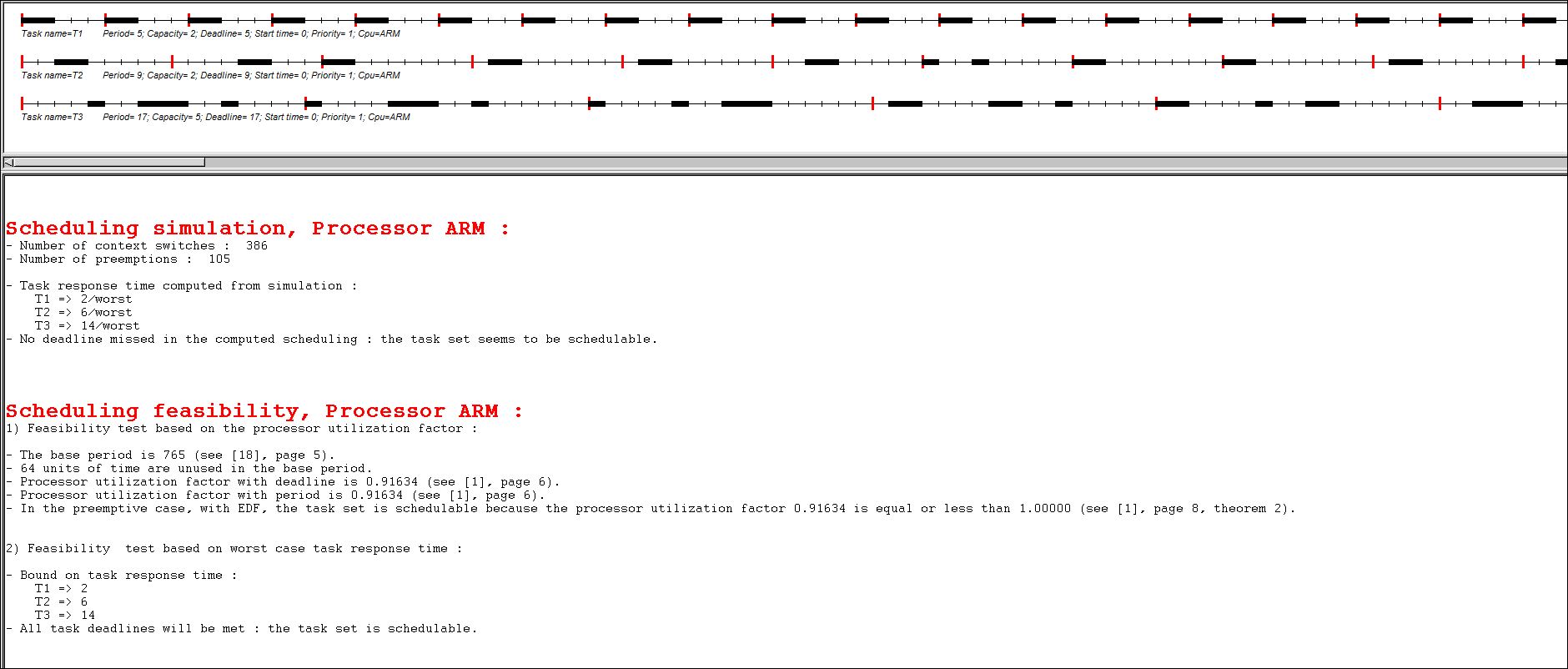
*Fig 7.3* ***Feasible****(schedulable) Example 7 using LLF*

**Example 8:**

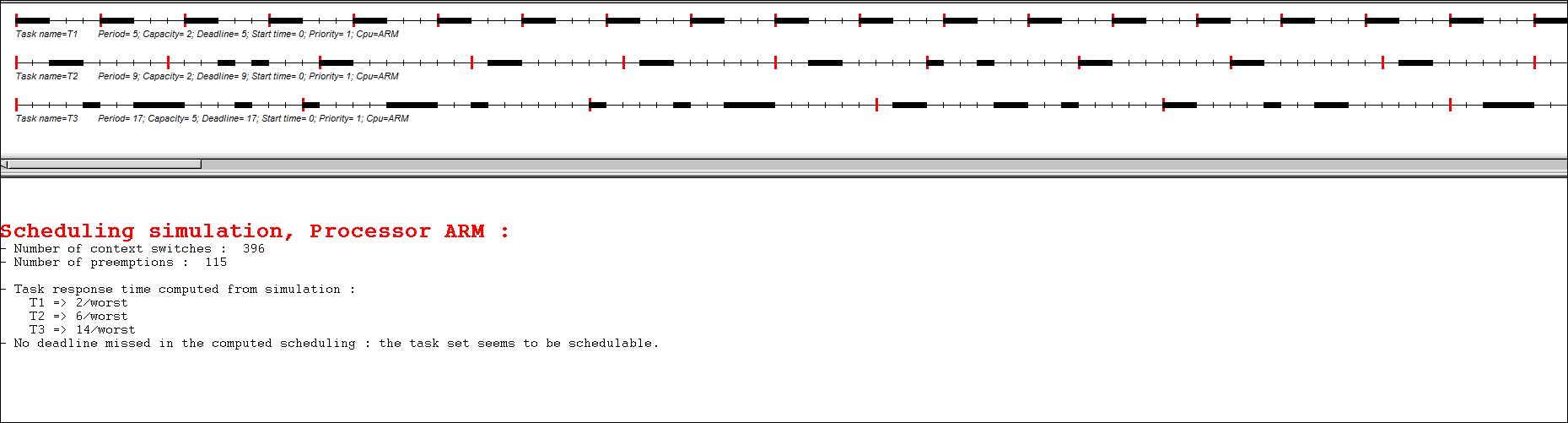
1. This set of tasks are schedulable according to cheddar using all policies
2. As this example wasn’t discussed as part of examples it has been compared by hand drawing timing diagram identified (Fig8.4) for maximum period of All tasks i.e. T3=17 (not for LCM=765).
3. While considering of scheduling it by hand for LLF (Fig 8.4), the laxity of S2 and S3 have is same (4). Assuming higher priority for a task having higher frequency of occurrence (S2) which matches with cheddar implementation (picking up a service randomly in this case as suggested by Prof/TA)



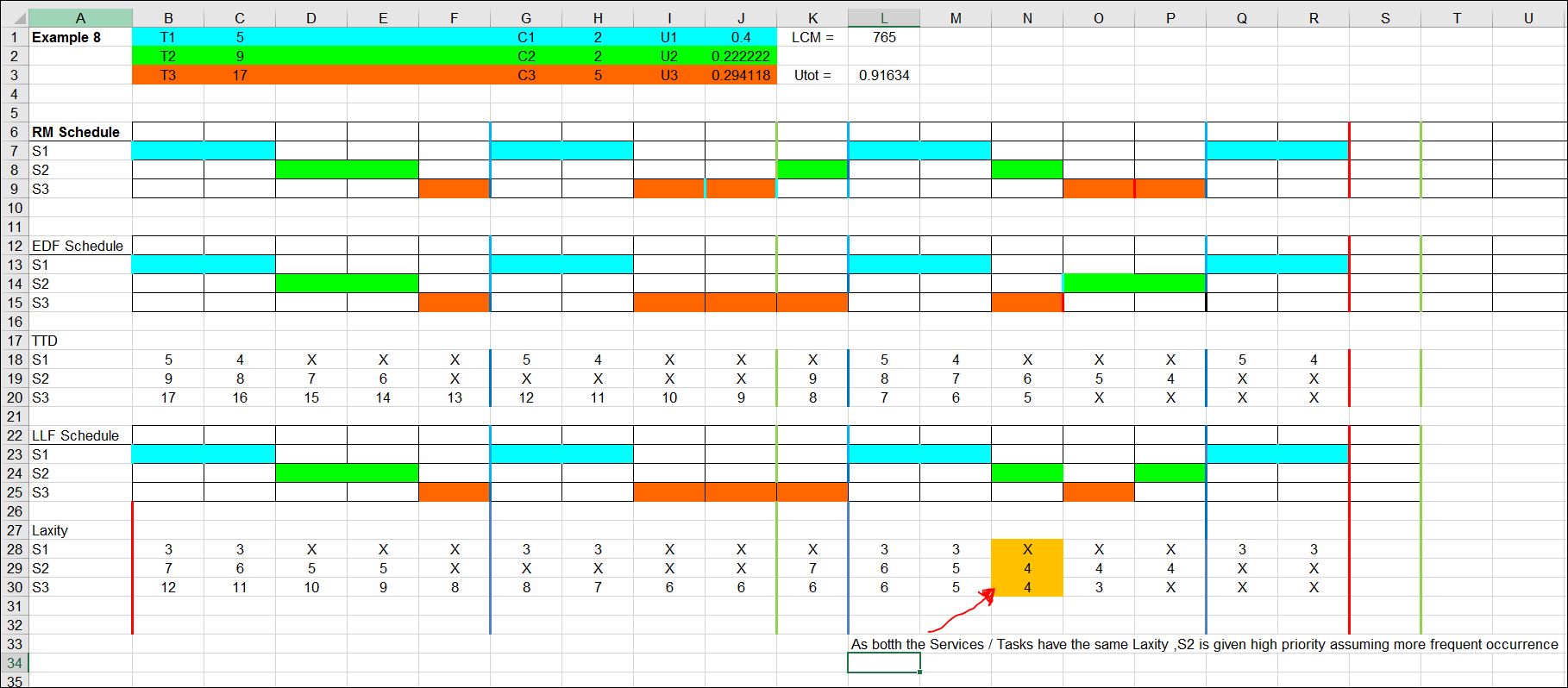
*Fig 8.1* ***Feasible****(schedulable) Example 8 using RMS*



*Fig 8.2* ***Feasible****(schedulable) Example 8 using EDF*



*Fig 8.3* ***Feasible****(schedulable) Example 8 using LLF*



*Fig 8.4**Timing Diagram for All policies for EDF, RMS and LLF for this task set*

*\*\* Complete Excel sheet for Ex 6 and 8 has been submitted within Q3 folder*

**Ans4.**

According to Lui and Layland Paper [1] and RTES book [2] the three **constraints** on RMS LUB are as follows:

1. Deadline of task is equal to period of occurrence of task.
   1. This in turn restricts multiple occurrences/instances of same service/task i.e. a task cannot occur within period of its own execution.
2. The tasks are preemptive and could be scheduled with a fixed priority for all time/mode of operation.

In-addition to it is run to completion scheduling.

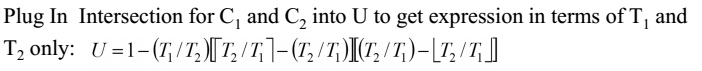
1. Tasks do not share any resources other than CPU.
2. All of the tasks require resources at the same time leading to worst case. The Critical instant causes maximum interference for lowest priority task/service and

RMS policy has three key **assumptions** are:

1) The requests for all tasks for which hard deadlines exist are periodic, with constant interval between requests.  
2) Deadlines consist of run-ability constraints only--i.e., each task must be completed before the next request for it occurs.  
3) Any non-periodic tasks in the system are special; they are initialization or failure-recovery routines; they displace periodic tasks while they themselves are being run, and do not themselves have hard, critical deadlines  
4) No Phasing. Each task is independent of each other and there are no dependencies to start and stop on each other. For example, event 1 and event 2 are not synchronous (related to each other) on time axis.

**Three key Derivation steps (tricky steps) that are important to understand**

1. (i)Combining equations for case 1 and case 2 at intersection point and replacing execution time C1 in terms of T1 and T2. This helps to obtain complete equation in terms of time periods (in this case Deadlines) independent of resources availability.

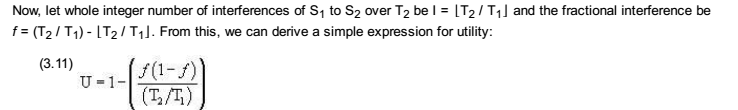


*Fig 9.1 Replacing C1 in terms of T1 and T2*

(ii)Thus the utility is a function of **disharmony or Harmony** between T2 and T1 (tasks scheduled). As the disharmony of the tasks increases **Utility decreases.**

(iii)It also proves that if tasks are perfectly harmonic and schedulable, CPU will be completely utilized i.e. U=1.

1. (i)Expressing a Utility Function, U (Fig9.1) in terms of f – fractional interference and I –Integer (floor of T2/T1) (Fig9.2)



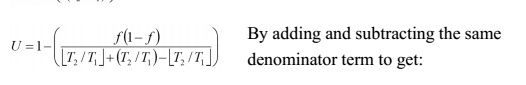
*Fig 9.2 Utility as a function of fractional (f) and Integer(real) value of T2 and T1 (source from Lecture 4 pdf)*

(ii)Further while deriving Fig 9.2 equation, [1] uses the understanding of the following math will be true for the model we are trying to derive. It comes with an additional condition of being f is non-zero. This is true as we are deriving for non-harmonic tasks (U! = 1) for non –ideal cases.

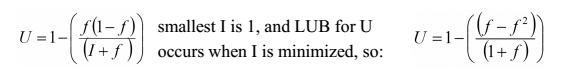


*Fig 9.3 Algebraic Math Equation*

1. (i) Adding and subtracting of floor value of (T2/T1) to denominator of Fig9.2 equations and converting to a left term equation of Fig 9.5. This represents terms in **I and f** as mentioned in point (2).



*Fig 9.4 Algebraic Math addition and subtraction of same term*



*Fig 9.5 Value of I=1 (for minimum U)*

(ii)Second tricky part before finding the max (upper bound) is replacing **I to 1.** According to [1] and [3] it is a Least upper bound derivation and worst value of **I** can be greater than 1. Further being in denominator is subtracting lesser value as it decreases. Thus worst case would be of **I=1(**smallest which can occur as I=floor(T2/T1)**).**

While understanding derivation, the mathematical part reiterates that it(math) enforces the upper bound limit to RMS LUB.

**References**

[1] Lui and Layland Paper <http://ecee.colorado.edu/~ecen5623/ecen/rtpapers/archive/PAPERS_READ_IN_CLASS/liu_layland.pdf>

[2] Real Time Embedded Components and Systems by Sam Siewert and John Pratt

[3] Lecture pdf 3 and 4

[4] paper "Architecture of the Space Shuttle Primary Avionics Software System"

[5] http://www.douglocke.com/Downloads/CylicvPriorityExecs.pdf