

# ***EDF SCHEDULER IMPLEMENTATION REPORT***

***Prepared by: Tarek Wael***

## **Abstract**

**Earliest Deadline First (EDF)** is an optimal dynamic priority scheduling algorithm used in real-time systems. It can be used for both static and dynamic real-time scheduling.

EDF uses priorities to the tasks for scheduling. It assigns priorities to the task according to the absolute deadline. The task whose deadline is closest gets the highest priority. The priorities are assigned and changed in a dynamic fashion. EDF is very efficient as compared to other scheduling algorithms in real-time systems. It can make the CPU utilization to about 100% while still guaranteeing the deadlines of all the tasks.

EDF includes the kernel overload. In EDF, if the CPU usage is less than 100%, then it means that all the tasks have met the deadline. EDF finds an optimal feasible schedule. The feasible schedule is one in which all the tasks in the system are executed within the deadline. If EDF is not able to find a feasible schedule for all the tasks in the real-time system, then it means that no other task scheduling algorithms in real-time systems can give a feasible schedule. All the tasks which are ready for execution should announce their deadline to EDF when the task becomes runnable.

EDF scheduling algorithm does not need the tasks or processes to be periodic and the tasks or processes require a fixed CPU burst time. In EDF, any executing task can be preempted if any other periodic instance with an earlier deadline is ready for execution and becomes active. Preemption is allowed in the Earliest Deadline First scheduling algorithm.

### *1. EDF Scheduler Implementation:*

**Task 1:** "Button\_1\_Monitor", {Periodicity: 50, Deadline: 50, E:18us}

**Task 2:** "Button\_2\_Monitor", {Periodicity: 50, Deadline: 50, E:18us}

**Task 3:** "Periodic Transmitter", {Periodicity: 100, Deadline: 100, E:42us}

**Task 4:** "UART Receiver", {Periodicity: 20, Deadline: 20, E:20us}

**Task 5:** "Load\_1\_Simulation", {Periodicity: 10, Deadline: 10, E:5ms}

**Task 6:** "Load\_2\_Simulation", {Periodicity: 100, Deadline: 100, E:12ms}

- *Execution times were calculated using GPIO trace hooks.*

### *2. Analytical Calculations:*

- Hyper period (**H**) = 100 ms (Highest deadline period).

- CPU Load (**U**) = Total Execution Time / Hyper period = (0 + 0 + 0 + 0 + 5\*10 + 12)

U= 62 %

### 3. Stimulability check:

#### 1- Rate monotonic method

$$U = \sum C_i / P_i \leq n * (2^{1/n} - 1)$$

where {n: number of tasks, C: execution time, P: periodicity}

$$U = 0/50 + 0/50 + 0/100 + 0/20 + 5/10 + 12/100 = 0.62$$

$$URM = 6 * (2^{1/6} - 1) = 0.73$$

$URM > U \rightarrow \therefore$  System is feasible

#### 2- Time demand analysis

- $W_i = e_i + \sum (t/P_k) * e_k$

#### Task 1 & 2

$$W(1) = 0 + (1/10) * 5 + (1/20) * 1 = 1$$

$$W(5) = 0 + (5/10) * 5 + (5/20) * 1 = 3$$

$$W(10) = 0 + (10/10) * 5 + (10/20) * 1 = 5$$

$$W(20) = 0 + (20/10) * 5 + (20/20) * 1 = 10$$

$$W(30) = 0 + (30/10) * 5 + (30/20) * 1 = 15$$

$$W(40) = 0 + (40/10) * 5 + (40/20) * 1 = 20$$

$$W(50) = 0 + (50/10) * 5 + (50/20) * 1 = D = 25 < 50$$

#### Task 3

$$W(1) = 0 + (1/10) * 5 + (1/20) * 0 + 2(1/50) * 0 = 1$$

$$W(20) = 0 + (20/10) * 5 + (20/20) * 0 + 2(20/50) * 0 = 10$$

$$W(50) = 0 + (50/10) * 5 + (50/20) * 0 + 2(50/50) * 0 = 25$$

$$W(100) = 0 + (100/10) * 5 + (100/20) * 0 + 2(100/50) * 0 = 50 < 100$$

### **Task 4**

$$W(1) = 0 + (1 / 10) * 5 = 1$$

$$W(3) = 0 + (3 / 10) * 5 = 2$$

$$W(5) = 0 + (5 / 10) * 5 = 3$$

$$W(7) = 0 + (7 / 10) * 5 = 4$$

$$W(9) = 0 + (9 / 10) * 5 = 5$$

$$W(11) = 0 + (11 / 10) * 5 = 6$$

$$W(13) = 0 + (13 / 10) * 5 = 7$$

$$W(15) = 0 + (15 / 10) * 5 = 8$$

$$W(17) = 0 + (17 / 10) * 5 = 9$$

$$W(19) = 0 + (19 / 10) * 5 = 10$$

$$W(2) = 0 + (2 / 10) * 5 = 1$$

$$W(4) = 0 + (4 / 10) * 5 = 2$$

$$W(6) = 0 + (6 / 10) * 5 = 3$$

$$W(8) = 0 + (8 / 10) * 5 = 4$$

$$W(10) = 0 + (10 / 10) * 5 = 5$$

$$W(12) = 0 + (12 / 10) * 5 = 6$$

$$W(14) = 0 + (14 / 10) * 5 = 7$$

$$W(16) = 0 + (16 / 10) * 5 = 8$$

$$W(18) = 0 + (18 / 10) * 5 = 9$$

$$W(20) = 0 + (20 / 10) * 5 = 10 < 20$$

### **Task 5**

$$W(1) = 5 + 0 = 5$$

$$W(2) = 5 + 0 = 5$$

$$W(3) = 5 + 0 = 5$$

$$W(4) = 5 + 0 = 5$$

$$W(5) = 5 + 0 = 5$$

$$W(6) = 5 + 0 = 5$$

$$W(7) = 5 + 0 = 5$$

$$W(8) = 5 + 0 = 5$$

## Task 6

$$W(1) = 12 + (1/10) * 5 + 0 + 0 + 0 = 12.5$$

$$W(2) = 12 + (2/10) * 5 + 0 + 0 + 0 = 13$$

$$W(10) = 12 + (10/10) * 5 + 0 + 0 + 0 = 17$$

$$W(20) = 12 + (20/10) * 5 + 0 + 0 + 0 = 22$$

$$W(50) = 12 + (50/10) * 5 + 0 + 0 + 0 = 37$$

$$W(70) = 12 + (70/10) * 5 + 0 + 0 + 0 = 47$$

$$W(100) = 12 + (100/10) * 5 + 0 + 0 + 0 = 62 < 100$$

***∴ System is schedulable***

4.Simso Simulator:

SimSo: Real-Time Scheduling Simulator

FileViewHelp

Gantt

Results

\* Unsaved

Qt

Model data

General

Scheduler

Processors

Tasks

id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)
1	TASK T1	Periodic	<input type="checkbox"/> No	0	50	-	50	0.014
2	TASK T2	Periodic	<input checked="" type="checkbox"/> No	0	50	-	50	0.014
3	TASK T3	Periodic	<input type="checkbox"/> No	0	100	-	100	0.018
4	TASK T4	Periodic	<input type="checkbox"/> No	0	20	-	20	0.024
5	TASK T5	Periodic	<input type="checkbox"/> No	0	10	-	10	5
6	TASK T6	Periodic	<input type="checkbox"/> No	0	100	-	100	12

<

>

Edit data fields...

Remove selected task(s)

Add task

Generate Task Set

Qt

Results

General

Logs

Tasks

Scheduler

Processors

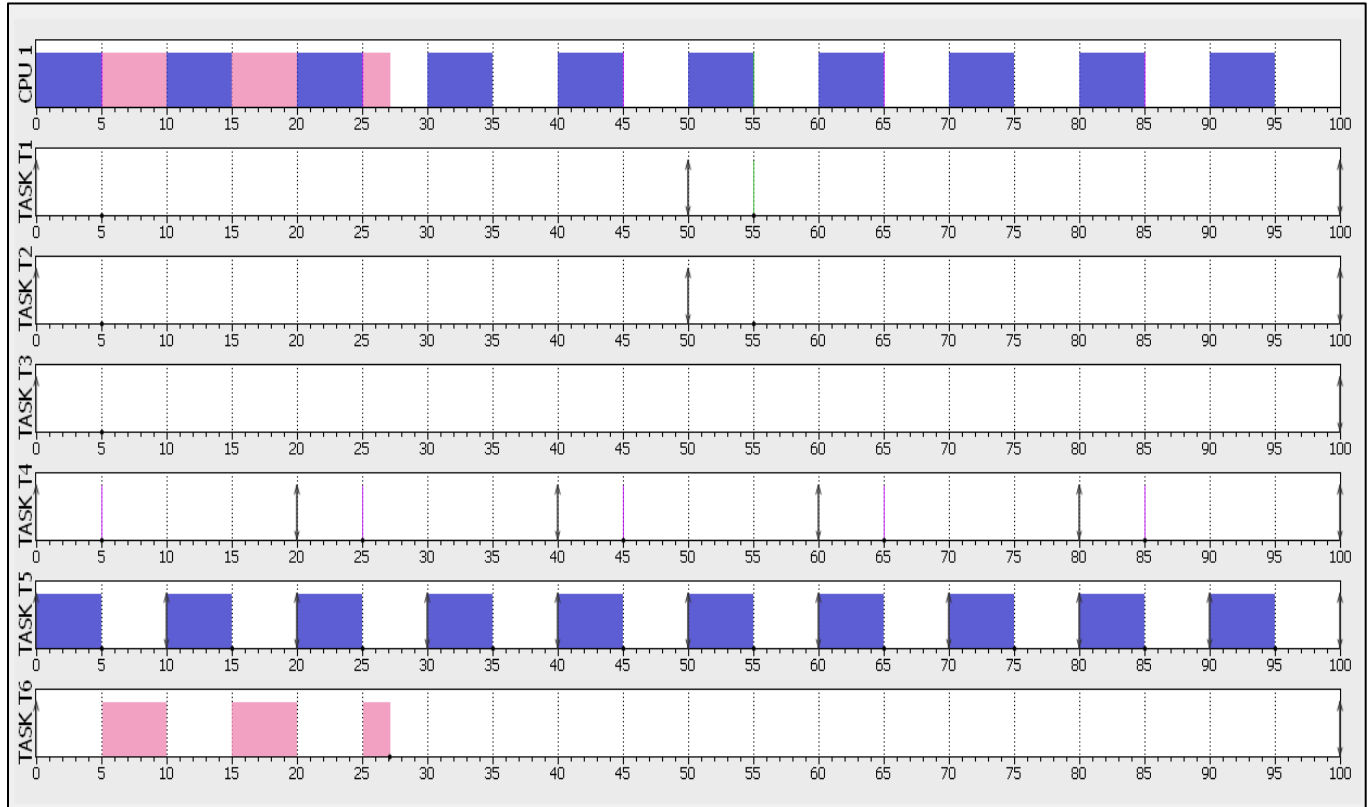
Observation Window:

from 0.00 to 100.00 ms

Configure...

	Total load	Payload	System load
CPU 1	0.6219	0.6219	0.0000
Average	0.6219	0.6219	0.0000

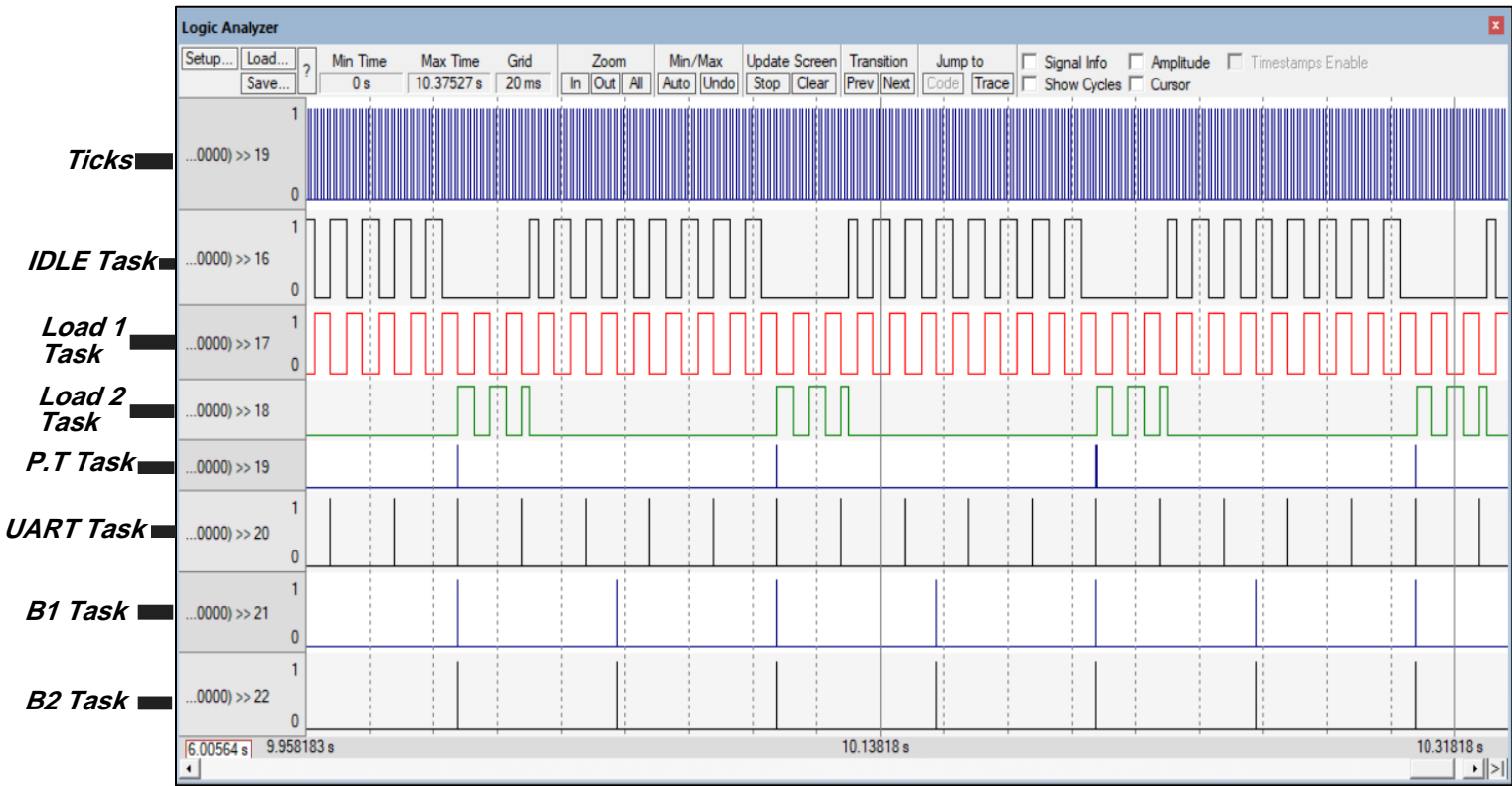
**Grantt Chart:**




- The Grantt Chart shows that the system is feasible, and no task breaks their deadlines.



5.Keil Simulator:



**- CPU Load = 62 %**

Name	Value	Type
 cpu_load	62	int
<Enter expression>		

- Used 3 queues to communicate between tasks.

[illegible]

- The figure above shows the rising (+) and falling (-) edges by setting and clearing bits 16, and 17 in port 1