International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

Scheduling of Dynamically Generated Tasks to find Energy Efficient Clock-cycle in CPU

Mrs. Bhuvaneshwari S Patil[1], Sheta Vipul B[2]

[1]Asst.Professor, Dept. of Computer Science and Engineering , NMIT, Bangalore, bsp14052001@gmail.com

[2]PG Student, Dept. of Computer Science and Engineering ,NMIT,Bangalore,vipul.sheta@gmail.com

Abstract— In computing field, central processing unit (CPU) plays role of running the jobs. The aim of CPU is to handle the jobs with respect to the scheduling methods of operating system. Jobs are being generated and executed which also allow preemption during running. Different numbers of scheduling methods already exist and many are in progress. In usual scheduling, there are chances of jobs not done, so completion of given set of jobs on time basis has created. Some techniques are used to find total possible jobs done along with its utilization. Concept used is smallest period and fastest deadline and some variation to complete the set of jobs. One of the concepts of green computing is to minimum power usage in CPU with keeping the clock cycle fixed .Energy usage is critical part of any computer systems. This can be used in any devices which run through battery e.g. mobile, laptops, embedded system, servers, etc. with respect to this project which will be mostly used for servers, where jobs are never ending. At operating system position, scheduling has being executed at different clock cycle and calculated energy used during their run time. Hence comparisons are made and analysis is done to find the energy efficient clock cycle with respect to fixed set continuous amount of jobs. The analysis involves the approximately constant time which is important criteria. Conclusion is derived which says "What Megahertz (MHz) of the CPU will perform jobs and save energy efficiently?"

Keywords—Scheduling ,CPU utilization,clock-cycle,preemptions,deadline,jobs,rate,monotonic,frequency,energy.

INTRODUCTION

Power consumption is important issue in battery held computing devices as it has limited energy. Using devices with full power and 100 percent CPU power will result in excessive heat and high electricity cost and even lifetime of that device decreases. Then again to cool those devices extra energy is used. This all leads to waste of resource. Thus question of reliability arises. In software side, the best way is to use different scheduling techniques and to choose the best according to device in which the software is going to be used. It's the combination of hardware and software, which plays far better role, rather than going alone. As per research much of energy is used by CPU rather than all other hardware like screen, cooling fan, speaker, etc. because workload is on CPU only. Because in nature, energy is limited and the usage in limited battery devices should be researched and response of all jobs should be done carefully. One method which is most common is dynamic voltage scaling is already playing important role. But here static clock-cycle and dynamic is task that is running. Here change in frequency at various times is done but kept constant till the jobs are completed.

GOALS

1. Find usage and availability of jobs.

Here, consideration of utilization factor of jobs is done which is from 0 to 100. With respect to this factor, possibility of jobs that would occur at various level of utilization needs to be count. This count is done in parallel with respect to all scheduling algorithm. The number will be displayed at each level of usage which shows amount of jobs that can be run and its value decreases as usage increases.

2. Fixing the clock-cycle and count the time.

With the help of a tool, clock-cycle values are being kept constant for entire period of execution. It is also being cross verified with the help of another tool before and after the execution of program. Time is being calculated through by default function in ubuntu. The consideration of user time is done because this is actual time taken by execution of process. The real and system time is not useful in the consideration. Hence they are ignored.

3. Count the energy used.

Energy calculation is based on amount of energy before execution of process minus amount of energy after execution. The result is termed as energy count for that specific process. Energy count also includes energy consumption of various hardware components but solution for that is also provided.

4. Compare the time and energy for respective clock-cycle.

This is analysis part which is being done by setting the frequency fixed and calculates energy and time factor for entire period of execution. Multiple times such value is counted in order to find energy efficient clock-cycle.

SURVEY

Energy is defined as multiplication of power and time at any instant [1].

$$E = P * t \quad \text{----------(1)}$$

E=energy consumption during execution.

P=power consumption

t = total time taken for execution

Depending upon task running or not, we have two types:

1 Dynamic: when CPU have task inside and power are used.

2. Static: when CPU don't have task, still power is being used.

Static scaling using rate monotonic and earliest deadline first gives better output and even all deadline are achieved [5].

Hard real time scheduling: It can be done in offline way to find worst case execution time (WCET) in task [6]. This method is definitely going to complete all jobs before their individual deadline.

Power can be estimated with respect to various simulations of jobs. Fixing the CPU clock-cycle values lead to more saving of energy. Balancing both factor that is of timeliness and energy is also important. Hence the system with variation of voltage levels is used to make experiment. For implementing these, set of task for execution is used but without pre-emption [8]. Starting with low energy earliest deadline first, check out initially with lowest frequency in order to check whether jobs are completed within time. Now, comparison can be made for power used with respect to utilization of task which proves earliest deadline first (EDF) consumes high but utilization is also high. Thus allowing pre-emption during runtime will lead to energy saving [8].

Basic requirements in Scheduling

1] Types of scheduling.
2] Types of jobs handling.
3] Allowable conditions.
4] Methodology with respect to hardware

1] Types of scheduling:

First come first served [FCFS]: It simply queues processes in the order that they arrive in the ready queue. It can give lower throughput because longer task can hold CPU for long time. This further leads to long waiting time and arrival time.

Earliest deadline first [EDF]:It is a dynamic scheduling algorithm used in real-time operating systems to place processes in a priority queue. Whenever a scheduling occurs, the queue finds for the process closest to its deadline, which will be the next to be scheduled for execution. In this algorithm, one can use the gap of time between consecutive jobs and within a single job. Within that loose timing, if it is used it will result in great difference in actual consumption of voltage in CPU [3].In [5] importance is given to deadline, the reason is real time task, if task is not completed as per time, the output is declared as wrong or unwanted.

Shortest remaining time job first : Here, the scheduler arranges processes with the shortest estimated execution time remaining to be next in the queue. This requires estimations about the time required for a process to complete. If a shorter process arrives during another process execution, the currently running process may be interrupted, dividing that process into two separate computing blocks The same rate monotonic based method with a difference can be used to minimize the energy for periodic jobs. It is applicable on jobs which are real time hard [4]. Along with this, three approaches can be combined. One is for each job, providing individual speed. Achieving the task allocation done perfectly is second. Third is main that is rate monotonic statically.

Fixed priority pre-emptive scheduling [FPPS]: Here, assignment of fixed priority rank to every process is done, and the scheduler arranges the processes in the ready queue as per their priority. Lower-priority processes get interrupted by new higher-priority processes.

Round-robin [RR]: The scheduler assigns a fix value time unit per process, and cycles continuously through them. It involves extensive overhead, more with a small time unit. As they have to repeat multiple times during cycle. Throughput here is shorter jobs are completed faster than in FIFO and longer processes are completed faster than in shortest job first. It has better average response time and waiting time depends on number of processes.

Specified level of assumptions: Here, whenever higher prior jobs enter, the smaller jobs have being given a threshold value for assumptions. Thus jobs can disable assumptions up to some level. So every job has a combination of priority assigned to them and addition to them this threshold value. Pre-emption is taking place with the condition that higher task has passed that value of threshold of executing job. Each jobs have their own priority, once they come into CPU for execution their priority is valued to their threshold. Hence non pre-emptive jobs can be controlled at much higher level. So, higher prior task can be held for some threshold period prescribed before start of execution [2]. It shows the advantage of decrease in overhead during execution.

Similar time for assumptions: In this algorithm, similar amount of time is provided to each jobs addition to their execution of period. This slot of time is equal among all set of jobs. Pre-emption is only allowed after that additional amount of time gets over for the smaller priority task. So this gives more time to low priority jobs.

Static assumptions: Here, whenever the jobs are running the pre-emption is not allowed at first. For introducing pre-emption some points are fixed at some locations before hand, so only at that points pre-emption is allowed, so set of jobs are divided into slots in a way that such slot do not allow pre-emption. If higher prior jobs occur during this slot, they have to wait till the slot is over or the fixed point comes. So here jobs need to give slot at some point.

2] Types of jobs handling:

There are devices which are not fully utilized and most of the time they are idle but the power utilization is constant throughout, thus need is to minimize it [1].

Periodic task is one that repeats itself after a certain fixed period interval. The precise time instants at which periodic tasks recur are usually limited by clock interrupts. For this reason, periodic tasks are sometimes called to as clock-driven tasks. The fixed time interval after which a task repeats is called the period of the task.

An aperiodic task can arise at random instants. However, in case of an aperiodic task, the minimum gap between two consecutive instances can be 0. That is, two or more instances of an aperiodic task might occur at the same time instant. Also, the deadline for aperiodic tasks is expressed as either an average value or is expressed statistically. Aperiodic tasks are soft real-time tasks.

3]Allowable conditions:

1. Whether to consider pre-emption or not?

When higher priority jobs arrive, the lower priority is stopped and higher is executed and rest of them are run at later.

2. Whether to migrate the running job or not?

It checks whether jobs can be changed during runtime from one processor to another.

3. Whether job can run on different processor simultaneously?

4] Methodology with respect to hardware:

DVFS method of reduction of energy can be used in many aspects. The real time of jobs cannot be counted because their execution takes separate path every time in inter task. In intra task the same job is being run, while change in frequency is done to minimize the time gap. During run time there is need of scaling up or down of voltage .Finding that point scaling point is aim to save energy [3].

Hardware and Software Interfaces

This project work is dependent on both hardware and software. The system should be run on battery operated laptop or notebook because that only will count total energy used. With different amount of processor power and ram power, the value of each scheduling and energy may vary. Rest all of above mentioned software should be installed without error.

Modules

1>header files: two header files are created for giving commonness of project parameters.

2>rest of main c program files are created.

3>on output side, we will check creation of pre-emption taking place in both of main algorithm.

4>second output generate total possibility of jobs created and usage.

5>analysis shows time factor and energy consumption during runtime of each output.

Diagrams

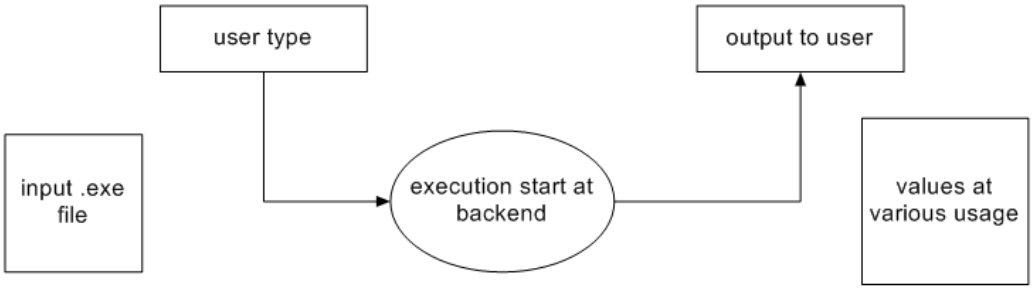


Fig:1 Dfd diagram level 0 for scheduling

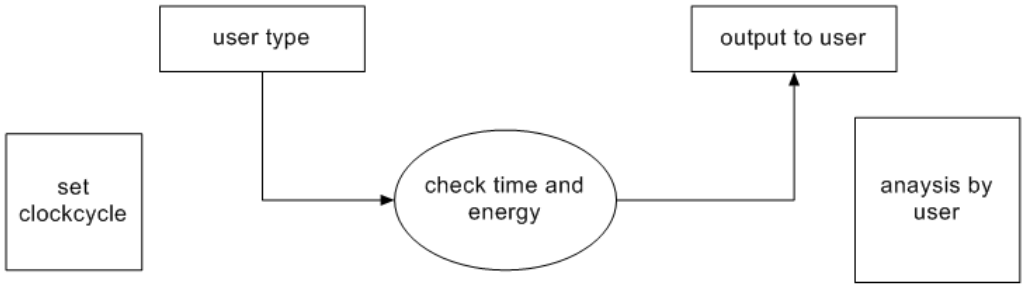


Fig:2 Dfd diagram level 1 for scheduling

Flow chart

Chart description: It is started with setting the clockcyle static for next process execution and checking the current energy level of the battery. In terminal, should be open and write the execution input. Here in this case it is .exe file. Now the process has started and various figures will start flowing on screen. The output will be time taken for execution and job possibility for all scheduling methods with respect to its usage. As soon as the execution is over, the energy is checked. On basis of it the analysis is done which contains counting of energy used.

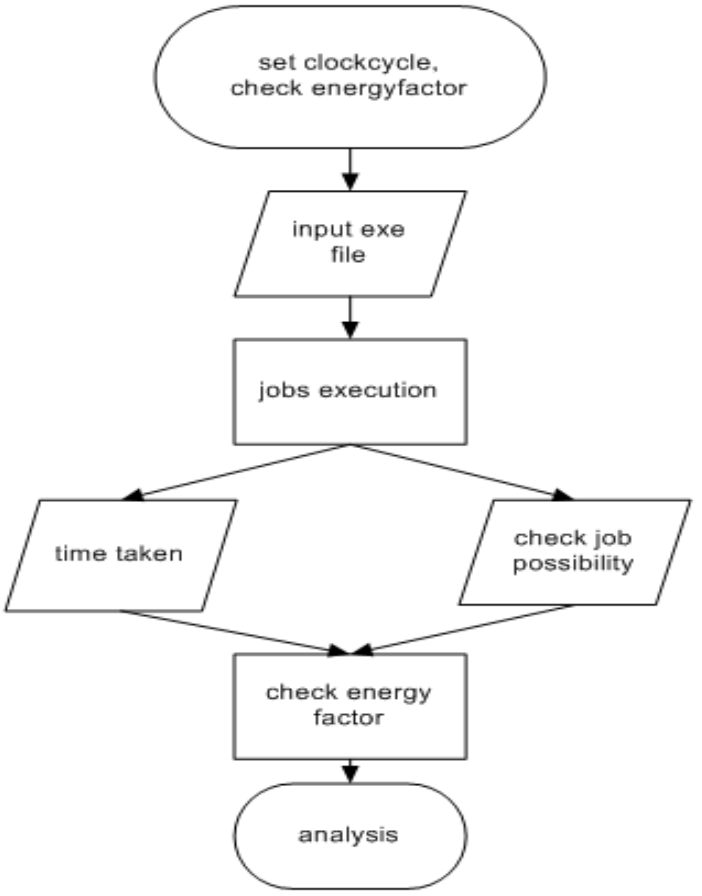


Fig:3 Flow chart for checking job possibility ratio

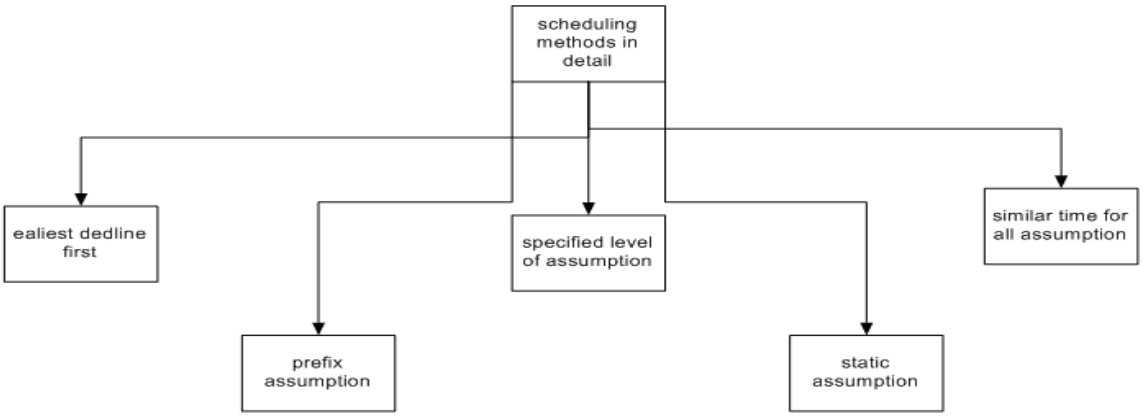


Fig:4 Block diagram for all scheduling techniques

Those are earliest deadline first which depends on deadline of the jobs, prefix assumption which depends on the period of completion of jobs, specified level of assumptions where at multiple level in execution the pre-emption is kept constant, static assumption where fixing the assumption is done and kept constant, similar time for all where each incoming jobs are given additional time of similar slot in order such that when some higher prior task is approaching it has to wait for that extra time.

PARAMETERS

1. Entire usage: the parameter is limited to proportion from zero to one hundred.
2. Cost: cost of assumption during the execution.
3. Task: total number of task.
4. States: all states in queue that is waiting, ready, current.
5. Prior: giving priority to job starting from zero.
6. Specifying level of assumption.
7. Worst case execution time.
8. Period of execution.
9. Deadline of jobs.
10. Arrival time of next job.

11 Remaining time of execution.

12. Allowance of pre-emption time.
13. Virtual time for job to be completed.
14. Similar time for all.
15. Static points for assumptions.

ALGORITHMS

Period and deadline based algorithm:

Steps:

1> Checking for possibility of process when pre-emption is allowed.
When the job is going to execute, the check is made for feasibility. So job can be said valid only if response time is before deadline period, if valid then job is counted in further step, else ignore.

2> Checking for possibility of complete set.
These check whether the entire set of jobs is feasible or not under pre-emptive conditions.

3> Counting of pre-emption done during execution time for rate monotonic.
It checks for virtual time is less than the execution time which is prescribed.

4> Insertion of process keeping factor deadline as main.
For EDF scheduling insertion of one by one task will take place under queue which is ready.

5> Keeping variable zero again.
Setting the variables pre-emption, virtual time of current job and last job has to reset to zero.

6> Entering the new upcoming process in pending line.
New job entering the execution needs to be in queue which is pending and also verifying that enqueue is ready.

7> Comparing the priority of current process and new coming process.

Comparison helps to decide for the pre-emption occur, if priority of new job is high than pre-emption will take place else will be ignored. Then the function puts the upper limit to algorithm. Again the check is made for possibility.

8> Getting response time and then checking for possibility of complete sets.

First the busy period of task is calculated and job is started. Response time of set is calculated in rate monotonic scheduling. Now the possibility of set is found with respect to rate monotonic.

Static assumption algorithm:

Steps:

1> Compute the function for float case.

The function is created to for calculating limit of task number which will be used in floating case. The limit is upper bound.

2> Calculating function by removing maximum chunk of task.

The new function is calculated referring to upper bound by removing the maximum chunk found in task.

3> Find active period of task.

The task remains busy for sum amount of time, that period is called active state. So for each task, the period is calculated as per their job number.

4> Find beta variable for non-float case.

Every task has to its blocking tolerance value, so it has to be found by keeping beta variable. For beta value equal to bound function along with the ending time.

5> Calculate chunk size of task.

Calculation of slot size is done for algorithm based on integer and possibility is checked. Also calculation of the floating value of chunk is done. Check for number of pre-emption taking place.

6> Check possibility of task.

For task with higher priority, the need of finding jobs is not required; requirement is chunk value and its block tolerance. Find worst case response time of job in order to predict possibility.

Specific level of assumption algorithm:

Steps:

1>Compare two task running and ready.

Initially, every job has increased its range to threshold described and thus pre- emption has its value higher than normal. Comparison is based on priority of current running job and upcoming new job.

2>Schedule the set.

The job set is scheduled again on basis of new threshold value, as increment is done to each job.

3>Block the low priority task

The task is blocked on basis of tolerance which occurred because of low priority task.

4>Calculate busy period.

It includes initial value of task starting with zero and new length of the task. If new length is bigger than the length is replaced with new length.

5>Start the jobs.

Job starting time is noted and loop is initiated to consecutive next jobs and finds lower bound of job.

6>Find worst case execution time (WCRT).

This step needs to check whether the job will complete in limited possible time, if not than job cannot be included.

7>Check the possibility.

Giving the least or lowest pre-emption threshold value and provide highest value to task in order to reduce the pre-emption.

RESULTS

Deadline and rate based algorithm preemption output.

USAGE	EDF	RM
50	9632	9794
55	11572	11789
60	13453	13726
65	15521	15880
70	17821	18277
75	19957	20527
80	22862	23526
85	26345	27078
90	30978	31802
95	36083	37356

Table1: total takeover of jobs vs. usage

These values changes with respect to type of processor. But the EDF will always has less number than rate monotonic (ratmo).

column	description
2	jobs with deadline based algorithm
3	static assumption with floating values
4	specified level of assumption
5	prefix assumption
6	similar time for all
7	static assumption with integer values

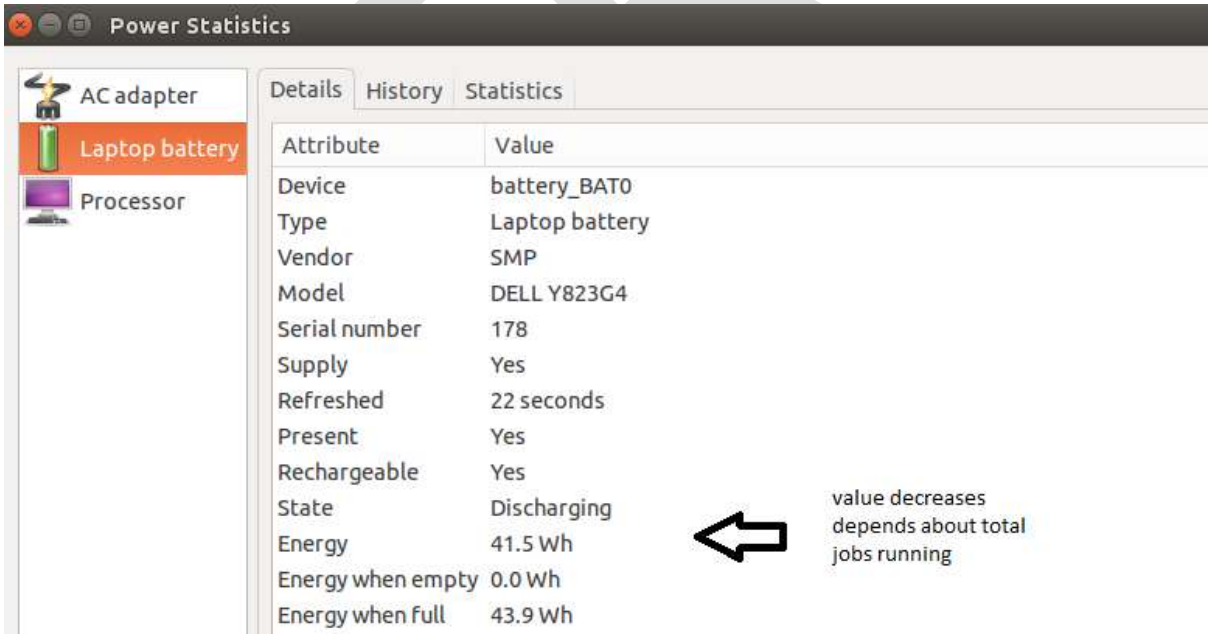
Table 2: Description details of table3

Scheduling of six algorithm ratio output

TOT USAGE	FASTDF	FIXFLPR	PREMTH	FIPRS	NONPRER	FIXFLPRINT
60	5000	5000	5000	5000	4356	5000
63	5000	5000	5000	5000	4261	5000
66	5000	5000	5000	5000	4088	5000
69	5000	5000	5000	5000	3887	5000
72	5000	5000	5000	4998	3617	5000
75	5000	5000	5000	4993	3303	5000
78	5000	4999	4996	4956	2837	4999
81	5000	4981	4958	4821	2260	4981
84	5000	4927	4803	4436	1427	4927
87	4997	4688	4315	3696	754	4688
90	4996	4008	3157	2353	247	4009
93	4977	2524	1516	1032	26	2524
96	4903	661	200	126	0	663
99	4234	3	1	1	0	3

Table 3: shows total usage of jobs vs. ratio of total possible job

Energy count



Here in above figure, energy is **41.5Wh** .That value decreases as usage is done by complete system hardware and jobs going on through some amount of time. Now say for e.g. while running the system constant for 2167 and do not work than also the system will be power drain in 1 hour 32 minutes approx.

But the same condition for 1000 MHz, duration longs with 2 hours 40 minutes. The difference can be seen in values itself. So this project analysis tells which frequency is optimum of all four frequencies which will satisfy both our situation of energy saving and good scheduling with good number of pre-emption.

The factor considering is both above stated, along with it is time factor. The frequency 1334 MHz here plays great role in completion of jobs and time factor. This is based on analysis.

Analysis

Frequency (MHz)→		1000		**1334**		1667		2167	
User time	Energy (W h)	23m 16.808s	5.8	**17m 29.844s**	**6.2**	15m 15.776s	6.3	10m 44.548s	7.2
		23m 25.611s	5.7	**17m 21.645s**	**6.1**	15m 23.568s	6.4	10m 48.680s	7.4
		23m 11.332s	5.7	**17m 36.237s**	**6.0**	15m 21.322s	6.4	10m 29.988s	7.3
		23m 9.221s	5.8	**17m 26.788s**	**6.1**	15m 29.462s	6.2	10m 32.365s	7.2

Table 4:Analysis for time taken and energy used for result two methods

Frequency(MHz)→		1000		**1334**		1667		2167	
User time	Energy(W h)	1m5.639s	0.4	**49.252s**	**0.3**	39.276s	0.4	30.232s	0.4
		1m5.532s	0.5	**49.332s**	**0.3**	39.332s	0.4	30.240s	0.4
		1m5.588s	0.4	**49.388s**	**0.3**	39.348s	0.4	30.226s	0.5
		1m5.488s	0.4	**49.364s**	**0.3**	39.326s	0.3	30.261s	0.5
		1m5.511s	0.4	**49.259s**	**0.3**	39.330s	0.4	30.278s	0.5
		1m5.610s	0.4	**49.271s**	**0.3**	39.359s	0.4	30.271s	0.4
		1m5.618s	0.5	**49.322s**	**0.4**	39.298s	0.4	30.220s	0.4

Table 5: Analysis for time taken and energy used for all scheduling methods

Here, the **1334 MHz** is best at long time execution of program. Differentiation with energy factor 6.2 of 1334 MHz and 6.3 of 1667 MHz. In general, as all other component is also working other than CPU, those parts also draining energy in equal quantity for longer time. So for first frequency the parts take more energy than the second frequency. The reason is longer time is taken by the first one.

The following example proves the point.

The overall components energy usage other than CPU is 1Wh per second. So for time 15 seconds, it will be 15Wh. For 30 seconds it will be 30Wh. So its components energy level is higher, while CPU energy level is gradually increasing at its own rate.

ACKNOWLEDGMENT

I extend my complete credit to my project guide, Mrs. Bhuvaneshvari S Patil, Asst. Professor ,Dept of CSE (PG), for helping me throughout the project work. She provided me with complete direction regarding the project and supported during entire period for completion of my project.I thank the complete teaching and non-teaching staff for their support and help.

CONCLUSION

The scheduling methods described here can be used in real time processing. The new methods described can be applied to set of jobs and can be executed for successfully completion of jobs efficiently. These methods as explained can run on CPU where there is continuous amount of jobs coming over time. The frequency which came as output can be considered more energy and time efficient. Even though basic method based on deadline and period are mostly used by making changes and updating, the new methods described can also be used to get better output and make flow of process in best possible manner. Finally it will ensure long lifetime in battery operated laptops or devices and decrease the consumption of energy in alternate current running devices. So the advantage of this work is great amount of energy savings in devices which have a varied range of frequency. Long time availability of device, so more work can be done for same amount of energy.

REFERENCES:

1] Power Aware Scheduling Algorithm for Real-Time Tasks over Multi-Processors, Muhammad Zakaiya, Uzma and Ayaz Ali Khan, Middle-East Journal of Scientific Research 15 (1),Islamabad, 2013.

2] Scheduling Fixed-Priority Tasks with Pre-emption Threshold, Yun Wang, Manas Saksena, RTCSA-IEEE, Hongkong, 1999.

3] Improving Energy-Efficient Real-Time Scheduling by Exploiting Code Instrumen- tation, Thorsten Zitterell and Christoph Scholl, Preprint from Proceedings of International Workshop on Real Time Software(RTS), Wisla, Poland, October 2008.

4] Energy-Aware Task Allocation for Rate Monotonic Scheduling, Tarek A. AlEnawy and Hakan Aydin, Real Time and Embedded Technology and Applications Symposium, 11th IEEE , 2005.

5] Power-Aware Scheduling in Embedded Systems: Adaptation of SDVS Scheduling
using Deadline Overrun Detection, Jonas Höglund, Stefan Olofsson, Magnus Persson, Johan Fredriksson, Detlef Scholle. Enea AB, Kista, First International Workshop on Energy Aware Design and Analysis of Cyber Physical Systems , Sweden, 2010.

6] Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time
Systems, Hakan Aydin, Rami Melhem, Daniel Mosse, Pedro Mejıa-Alvarez, Real-Time Systems Symposium,(RTSS) Proceedings. 22nd IEEE, 2001.

7] Algorithms for Energy Saving. Susanne AlbersS, H. Alt, and S. Naher (Eds.) Festschrift Mehlhorn, LNCS 5760, Springer, 2009.

8] Experiences in Implementing an Energy-Driven Task Scheduler in RT-Linux, Vishnu Swaminathan, Charles B. Schweizer, Krishnendu Chakrabarty and Amil A. Patel, Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium,2002.
9] Software or Hardware: The Future of Green Enterprise Computing, Maria Kazandjieva, Brandon Heller, Omprakash Gnawali, Wanja Hofer, Philip Levis, Christos Kozyrakis. Stanford university edu, California, 2011-12.

10] Fundamentals of Power-Aware Scheduling, Xiaobo Sharon Hu1 and Gang Quan2, J. Henkel and S.Parameswaran (eds.), Designing Embedded Processors A Low Power Perspective, Netherlands, 2007.

11] https://wiki.archlinux.org/index.php/CPU_frequency_scaling

12] http://www.makeuseof.com/tag/control-power-usage-gnome-power-statistics-linux/

13] Power-Aware Scheduling for Periodic Real-Time Tasks, Hakan Aydin, Daniel Mosse, Pedro Mejia-Alvarez, IEEE transactions on computers, 2004.

14] An Efficient Power-Aware Scheduling Algorithm in Real Time System, Hyekseong Kweon, Younggu Do, Jaejeong Lee, Byoungchul Ahn, PACRIM IEEE,2007.

15] Frequency-Utilization Based Power-Aware Schedule Policy for Real-Time Multi-cores System, Lin Zhou, Lei Yu, IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013