A Proposed Off-Line DVS Approach for Minimizing Energy Consumption

Suaiman M Abbas Yarub Y Abbas
Electrical Engineering Department / College Of Engineering / University Of Baghdad

ABSTRACT
This paper concerns with the issue of minimizing of ever increasing energy consumption in computer systems .The minimization of energy consumption can be achieved by help of hardware feature in the processors called Dynamic Voltage Scaling (DVS).When the applied voltage on processor decreased then the energy consumption of processor will decrease ,but at the same time the speed of the processor will also decrease .The decision of when and how much to change the applied voltage will be the job of the DVS algorithm.

One of the software methods to maximize energy saving is redistributing the available slack among the tasks .An important derivation of slack distribution among tasks says that when the processor execute all tasks with only one clock frequency then the energy consumption is minimized, and the proposed algorithm adapted this derivation in its kernel as optimum energy saving solution .This paper proposes an algorithm to solve the problem of deadline violation of tasks (the violation happen due to distribution of the slack among tasks) by new method which eliminates the amount of violation from the total slack and redistribute the updated slack again.

General Terms
Power minimization ,task scheduling

Keywords
Dynamic voltage scaling ,slack distribution ,power dissipation minimizin

1. INTRODUCTION
The rapid progress in semiconductor technology has led to higher chip density and operation frequency ,making today's systems more complex and power hungry [1].The power dissipation of modern processors has been rapidly increasing along with increasing transistor count and clock frequencies. At the same time ,there is growing disparity between maximum power consumption of a processor and the typical power consumed by that processor ,i.e. power consumed while running typical applications .This trend is the result of the significant increase in transistor count required to reach the desired peak performance targets[3].Power consumption is increasing because frequency and leakage current are scaling up so much that their effect on power cannot be offset .Such trend makes the cost of cooling system grow and challenges the performance benefits that can obtained by the ever-growing transistor density[3].There are several reasons for this new interest in dynamic voltage scaling(DVS).First, portable and handheld systems have limited battery usage ,so power saving will increase battery life. Second , a reduction in power consumption leads to increased reliability in microprocessors .Higher power devices dissipate more heat ,which can permanently damage a chip ,so lowering a processors power reduces the likelihood of failure .Third ,reducing power can reduce cost.

2. DYNAMIC VOLTAGE SCALING (DVS)
Dynamic voltage and frequency scaling is a mechanism whereby software (or operating system) can dynamically adjust central processing unit (CPU) voltage and frequency .This mechanism allows systems to address the problem of ever-increasing CPU power dissipation and energy consumption ,as they are both quadratically proportional to the CPU voltage .However ,reducing CPU voltage may also require CPU frequency to be reduced and results in degraded CPU performance with respect to execution time .In other words ,DVS trades off performance for power and energy reduction[4].In general ,CMOS circuits consume power proportional to V^2f where Vis the voltage and f is the frequency .Energy consumption per cycle is power consumption divided by frequency ,so energy consumption is proportional to V^2.In other words ,reducing the voltage quadratically reduces the energy needed to perform the same number of cycles .However ,a performance trade-off arises because running at a lower voltage increases gate settling time and thus necessities running at a lower frequency .The maximum valid frequency for a given voltage is roughly linear in voltage ,more accurately ,it is proportional to $(V-V_{th})^2/V$ where V_{th} is the threshold voltage of the CMOS processor .Due to this trade-off between performance and energy consumption ,the decision about when to raise or lower the speed is a complex one requiring knowledge about CPU requirements both now and in the future. DVS algorithms attempt to predict such requirements and adjust speed and voltage accordingly [5]

3. POWER AND ENERGY REDUCTION IN CMOS
Digital CMOS circuits are used in the industry of modern microprocessors .Power dissipation in CMOS circuits arises from three different mechanisms: static power ,which results from resistive paths connecting power supply to ground ,dynamic power which results from switching capacitive loads between two different voltage states[6],and the short-circuit power due to flow of I_S[7].The power of CMOS gate is simply approximated as:

$$POWER_{gate} = P_{dynamic} + P_{LEAK} + P_{SHORT}$$

$$= a.f.C.{V_{dd}}^2 + I_0.10^{\frac{Vth}{S}}.V_{dd} + a.I_S.V_{dd} \qquad (1)$$

where a is switching activity of the gate, f is the operating frequency ,C is the load capacitance ($C=C_G+C_J+C_{INT}$, where C_G ,C_J ,and C_{INT} denote gate ,junction ,and interconnection capacitance respectively), V_{dd} is the supply voltage, I_O is the drain current when the threshold voltage is equal to zero, S is the device parameter, V_{th} is the threshold voltage of the transistor, and I_S is the current which flows through turning-off MOSFET .With relatively small static power and short-

circuit power ,charging and discharging capacitors generally consumes most of the power on CMOS circuit .The dynamic power of this gate(or node) is the energy per cycle times the number of cycles as shown:

$$P_{dynamic} = C \cdot V^2{}_{dd} \cdot a \cdot f \quad (2)$$

where a is the number of times this node cycles each clock cycle and is usually called the activity ratio .The dynamic power for the whole chip is the sum of last equation over all the nodes in the circuit [6].The dynamic power consumption for whole chip can be modeled by:

$$P_{dynamic} = \sum_{i=1}^{N} C_i \cdot f_i \cdot V^2{}_{dd} \quad (3)$$

Last equation can be simplified by assuming that all gates g_i create a collective switching capacitance operating at a common switching frequency f [8],thus

$$P_{dynamic} = \beta \cdot C \cdot f \cdot V^2{}_{dd} \quad (4)$$

Last equation shows that lowering the clock frequency linearly decreases power, but that voltage reduction results in a squared power reduction.

3.1 System-level Techniques for Energy Reduction

Energy management can be done at several levels in the computer system hierarchy: the component level ,the operating system level ,the application level ,and the user level .The end-to-end argument suggests that this management should be performed at the highest level possible ,because lower levels have less information about the overall workload .Most energy management is best performed at the operating system level.

Dynamic power management(DPM) and Dynamic voltage scaling(DVS) represent the two operating system-level techniques to reduce the system power consumption.DPM refers to the use of power-down modes when the processor is idle to reduce the processor power consumption.DVS refers to dynamically varying the speed of a processor by changing the clock frequency along with the supply voltage[1].

3.2 Deep investigation in time equation of CMOS processors

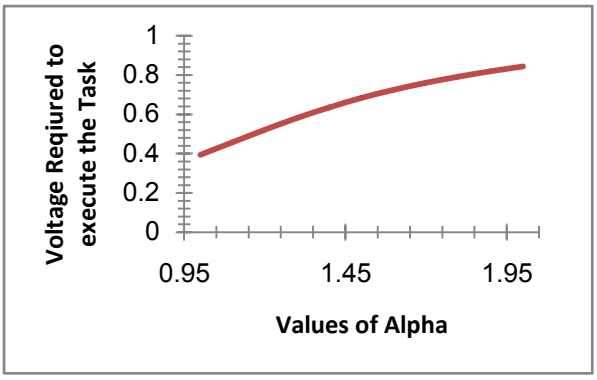
Decreasing V_{dd} will minimize the power consumption ,but on the other hand ,this will lead to maximizing propagation delay of the gates and hence decrease the overall processor performance ,so for this task ,the execution time T(V_{dd}) according to this equation will get longer:

$$T_j(V_{dd}) = NC_j \times K \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \qquad 1 < \alpha \leq 2 \quad (5)$$

Where $T_j(V_{dd})$ is execution time of task $_j$ under applied voltage V_{dd} , NC_J is required number of cycles to execute the task , V_{th} is gate threshold voltage, K,a are constants depending on CMOS processor characteristics .The aim of all DVS algorithms is to minimize V_{dd} as much as possible to save power, or to maximize the execution time of particular task without violating that task deadline .The algorithms should be able to determine the correct voltage V_{dd} and correct timing to use it .From last equation a formula can be found that gives the exact value of the voltage V_{dd} for a given (or calculated) value of $T(V_{dd})$ and other constant parameters (i.e. K,a,V_{th},C_L).For comprehensive view of this topic ,the variations of these parameters will be considered one by one.

3.2.1. A Variation

A tends to be decreased so that lower voltages can excite the logic gates faster. "Figure 1" shows this point for same job where V_j increases as a increase. The values of a is given by the formula 1<a<2, and that means there are three values of a namely: a=2, 1<a<2, a=1



Figure(1) : voltages needed to execute task with different α values.

3.2.1.1 Case a=2

the modern DVS-CMOS processors have a around 1.3 and hence consumes less power .So considering a=2 gives indication for maximum level of the voltage V_{dd} .For simplicity let V_{dd} =V_j which means the correct voltage for task$_j$,under these conditions last equation would be:

$$T(V_j) = NC_j \times K\,CL \frac{V_j}{(V_j - V_{th})^2} \quad (6)$$

(6)

It is clear that value of $K.C_L.V_j / (Vj - V_{th})^2$ represents the amount of time for one cycle under the voltage V_j.The derived voltage Vj from last equation gives this following real and practical solution:

$$V_j = 0.5 \times \left(2\,V_{th} + \frac{NCj}{T(Vj)}\,k\right) \mp 0.5 \sqrt{\left(2\,V_{th} + \frac{NCj}{T(Vj)}k\right)^2 - 4\,V_{th}{}^2} \quad (7)$$

3.2.1.2. Case a=1

that indicates ideal CMOS processor because the modern CMOS processors have a greater than one, equation[5] would be:

$$T(V_j) = NC_j \times K \frac{V_j}{(V_j - V_{th})^1} \quad (8)$$

The derived voltage V_j from last equation gives the following solution :

$$V_j = 0.5 \frac{V_{th}}{\left(1 - \frac{NC_j}{T(V_j)}\,k\right)^1} \quad (9)$$

Where

$$1 > \frac{NC_j}{T(V_j)}\,k > 0$$

3.2.1.3. Case 1<a<2

the typical and practical values of a are in the range between 1 and 2.The trend now to minimize a which will reduce propagation delay of gates ,and that means lower voltages and higher processor speed .That means minimizing dynamic power term of total power equation[1].Equation[5] becomes:

$$T(V_j) = NC_j \times K \frac{V_j}{(V_j - V_{th})^{\alpha}} , \qquad 1 < \alpha < 2 \qquad (10)$$

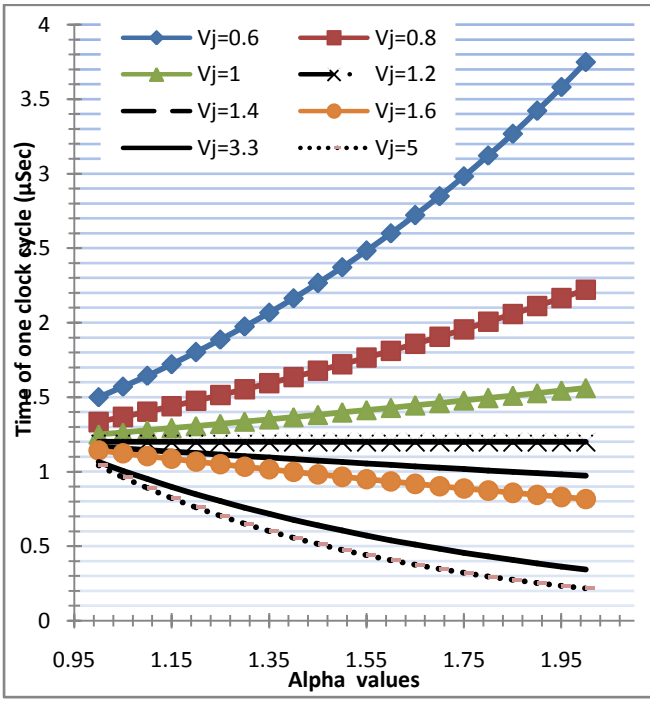
Solving above equation is not an easy task and it needs numerical methods to get an approximate value of V_j .Last equation can be reformulated as shown:

$$(V_j - V_{th})^{\alpha} - V_j \frac{NC_j \times K}{T(V_j)} = error \qquad (11)$$

The correct value of V_j can be approached by using last equation repeatedly until the difference becomes acceptable .The approach used to find acceptable error is important because bad computation lead to longer operation and hence more power consumption.

3.2.2. $(V_j - V_{th})^{\alpha}$ variation

As a greater than 1 and approaching 2, the magnitude (V_j - V_{th})^a will behave differently depending on the value of (V_j - V_{th}) compared to 1,mathematically speaking there are three possible variations of (V_j -Vth) namely:if (V_j –V_{th}) <1, a is minimized, close to 1, so it can keep the gates propagation delay time relatively short and overall processor speed high (i.e.T(V_j) short) as shown in figure[2] for V_j = 0.6,0.8.and1.0 the propagation delay increases as a increase .



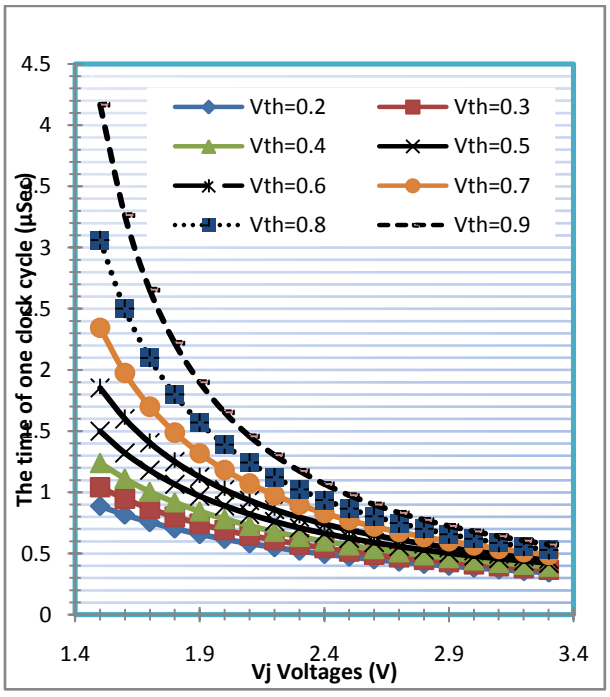
Figure(2): T(V_i) vs. α where α changes from 1 to 2 with Vth = 0.2 and six V_i values .

If (V_j -Vth) >1 ,it is better to maximize a to be close to 2 rather than 1 ,because when a=2 then T(V_j) is shorter than T(V_j) when a = 1 ,as seen from figure[2] for voltages V_j =1.4,1.6,3.3,and 5.It can be seen from last figure that voltages 3.3 and 5 achieve close processor speed ,so it is better to pick 3.3 over 5 ,to minimize energy consumption .If (V_j –Vth) =1,then the voltages do not depend on a as seen from figure .It can be noticed from last figure that the variation in cycle time get more and more extremer as a approaches 2 while it is less as a approaches 1.So for a system operating with different levels of voltages like DVS processor it is always recommended to minimize a to be close to 1 as possible so a will not has that influence on T(V_j) as shown in figure[2].It is also recommended for DVS systems to choose levels of operating voltages as near to (1+ V_{th})V as possible ,since in these levels neither value of a nor the number of operating voltage levels will have that impact on T(V_j) as shown in figure[2] for Vj =1,1.2,1.4,and1.6

3.2.3. Vth variation

It is obvious from equation[5] that V_{dd} should be greater than V_{th} ,because the logic gates will not be excited ,or mathematically speaking when V_{dd} = V_{th} the time of cycle T(V_{dd}) approaches infinity .Figure[3] shows that if V_j is slightly greater than V_{th} ,gates excitation will be very slow and vice versa. It can be seen that after a certain value of V_j , T(Vj) will not change that much, and that small values of V_{th} leads to faster operation for gates .Equation (5) becomes here:

$$T(V_j) = \frac{V_j}{(V_j - V_{th})^{\alpha}} \qquad 1 < \alpha \leq 2 \quad (12)$$



Figure(3) : T(V_I) vs. V_I with different values of Vth and α

4. OPTIMUM OFF-LINE DVS ALGORITHM

It is known that there are many DVS algorithms ,each one solve the scheduling problem in different method targeting particular problem .This paper proposed DVS algorithm concentrated on simplicity as a primary target to solve the scheduling problem ,because the algorithm itself consumes processor speed and energy .The approach used here is to convert the problem of finding correct voltage to mathematical model and simply substitute the variables to get the exact voltage as in equations[7,9,11].It is assumed that the target set of tasks have the same moment of existence(i.e.all tasks exist at time=0).If for certain reason another task exist at time>0,then the algorithm cancels the old optimal scheduling

and repeat the procedure of calculating optimal scheduling including the new tasks.

4.1 Scheduling condition for Tasks

Most of the operation systems arrange the tasks according to certain policy ,and gives priorities accordingly .For a set tasks to be schedulable ,the execution of all tasks at maximum processor speed should not violate any of their deadlines .If the set of tasks not schedulable then it is not possible to make any successful schedule .There is a schedulable condition for each single task ,so for a task to be schedulable among a set of tasks ,the execution of all higher priority tasks one by one at maximum processor speed including the execution of this task should not violate this task deadline .The above condition can be represented by mathematical formula as given:

$$\frac{\sum_{j=1}^{N} ET_j}{DL_N} \leq 1 \qquad (13)$$

The summation of execution times of all higher priority tasks include task N($\sum$ ET_j) should not violate the deadline DL_N OF TASK N .For a particular task x to be scehedulable , the following condition must be fulfilled :

$$\left(\frac{\sum_{j=1}^{x} ET_j}{DL_x} \leq 1 \qquad (14)$$

4.2 Minimum power scheduling algorithm for non-periodic tasks

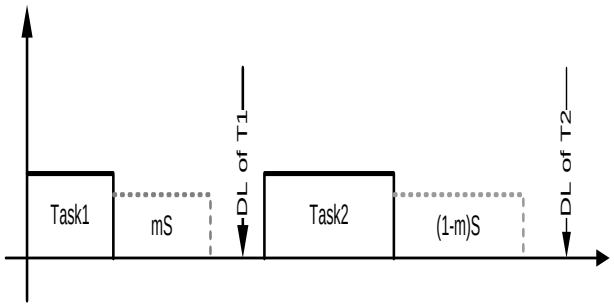
A set of non-periodic tasks is that set of tasks which are not repeated periodically ,but be considered as one period tasks .The proposed algorithm will find minimum power consumption for this kind of tasks .In the proposed algorithm ,it is assumed that there are N non-periodic tasks ,but before making any kind of scheduling, the algorithm has to make sure all N tasks must satisfy condition in equation[13,14].The tasks must be arranged and given priority according to certain policy .The most effective policy in DVS litertures is the earliest deadline first policy(EDF).If a set of tasks has feasible deadlines, then scheduling in increasing deadline order will always satisfy all deadlines .If the tasks passed the schedulable conditions ,then any available slack has to be found when executing all the tasks at maximum processor speed or not .To do that ,it is needed to find the total summation of execution times of all tasks altogether at maximum speed and subtract it from last priority task deadline .If the result is positive ,then there is a slack given by:

$$Slack = DL_N - \sum_{j=1}^{N} ET_j \qquad (15)$$

If the slack is greater than specified threshold then this slack worth distributing among tasks to save power consumption .Thus algorithm should distribute the slack among tasks without missing taskj deadlines.

4.3. Optimum slack distribution between two tasks

It is mentioned earlier when the amount of available slack is large enough(i.e. slack>threshold),then the distribution of the slack is worthy and important to minimize energy consumption .But this raises another problem about optimal method of distributing this slack .To show this point :assume that there are two tasks ,namely task1 and task2 with the number of execution cycles are NC_1 andNC_2 respectively and the available slack S.The problem is how to decide how much share of S would be given for each task to ensure minimum energy consumption as shown in figure[4].



Figure(4) : Two tasks with optimal slack distribution

It is known that for CMOS processors ,the power consumption is dominated by dynamic power dissipation P_d as in [9],

$$P_d = C_{ef} \cdot f \cdot V_{dd}^2 \qquad (16)$$

Now processor speed is almost linearly related to supply voltage for $V_{dd}>V_{th}$[9]

$$f = k \frac{(V_{dd} - V_{th})^2}{V_{dd}} \qquad (17)$$

thus P_d is almost cubically related to f [9]

$$P_d \approx C \cdot \frac{f^3}{k^2} \qquad (18)$$

assuming that the system is working at maximum processor speed or at $f_{max}=1/ T_{min}$, then power cosumed by such system is $P_d = C.f_{max}^{3,}$ which is true for all tasks, but the energy consumed by each task is different:

$$E_{Task\,2} = P_d \cdot T_2 \quad E_{Task\,1} = P_d \cdot T_1$$

where T_1,T_2 are execution time for task1 and task2 under clock frequency f_{max} .If the operating frequency changes for each task ,then the calculation would be different ,i.e. each task will dissipate different power and consume different energy as given:

$$P_1 = C \cdot f_1^3 , \quad P_2 = C \cdot f_2^3 \qquad (19)$$

where P_1 ,P_2 are power dissipation for $task_1$ and $task_2$ and the corresponding energy are given by:

$$E_1 = P_1 \cdot t_1 , \quad E_2 = P_2 \cdot t_2 \qquad (20)$$

where t1,t2 are the execution time of task1 and task2 at the frequencies f1and f2. Now, assuming the new task execution times after distributing slack S are:

$$t_1 = (T_1 + m.S), \quad t_2 = (T_2 + (1-m).S) \qquad (21)$$

m<1

the new cycle time for each task clock frequency:

$$cycle_1 = T_{min} + \frac{m.S}{NC_1} , \quad cycle_2 = T_{min} + \frac{(1-m).S}{NC_2} \qquad (22)$$

It is known that the execution time at f_{max} of each task is the time of one cycle times the number of cycles, i.e.

$$T_1 = NC_1 \,.\, T_{min} \;, \quad T_2 = NC_2 \,.\, T_{min}$$

where

$$NC_1 = \frac{T_1}{T_{min}} \;, \qquad NC_2 = \frac{T_2}{T_{min}} \qquad (23)$$

and

$$cycle_1 = T_{min} + \frac{m.S}{T_1/T_{min}} = \frac{T_{min}}{T_1} \,.\, (T_1 + m.S) \,,$$

$$cycle_2 = T_{min} + \frac{(1-m).S}{T_2/T_{Tmin}} = \frac{T_{min}}{T_2} \,.\, (T_2 + (1-m).S) \qquad (24)$$

substitute values from (24) into 19 ,where $f_1 = 1/\ cycle_1$,$f_2 = 1/\ cycle_2$,yields

$$P_1 = C.\frac{1}{\frac{T_{min}^3}{T_1^3}.(T_1 + m.S)^3} \,,$$

$$P_2 = C.\frac{1}{\frac{T_{min}^3}{T_2^3}.(T_2 + (1-m).S)^3} \qquad (25)$$

Since E = P.t ,then

$$E_1 = C.\frac{T_1^3}{T_{min}^3.(T_1 + m.S)^3} \,.\, (T_1 + m.S) = C.\frac{T_1^3}{T_{min}^3.(T_1 + m.S)^2} \;,$$

$$E_2 = C.\frac{T_2^3}{T_{min}^3.(T_2 + (1-m).S)^3} \,.\, (T_2 + (1-m).S) = C.\frac{T_2^3}{T_{min}^3.(T_2 + (1-m).S)^2}$$

The total energy consumption for both tasks is

$$E_T = E_1 + E_2 = \frac{C}{T_{min}^3}.\left[\frac{T_1^3}{(T_1 + m.S)^2} + \frac{T_2^3}{(T_2 + (1-m).S)^2}\right] \qquad (26)$$

Deriving last equation to find the minimum possible value gives:

$$\frac{d\,(E_T)}{dm} = 0 = \frac{-T_1^3}{(T_1 + m.S)^3} + \frac{T_2^3}{(T_2 + (1-m).S)^3}$$

which gives ,

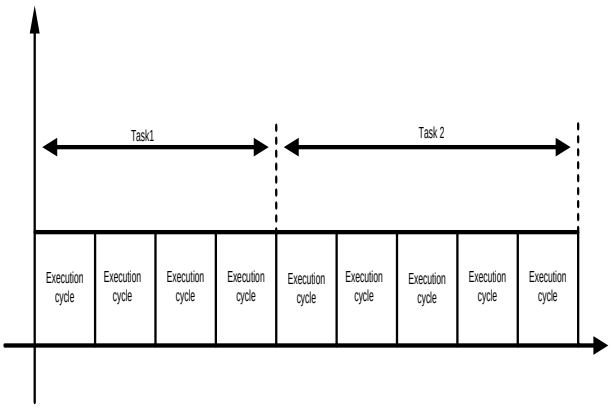
$$\frac{T_1}{(T_1 + m.S)} = \frac{T_2}{(T_2 + (1-m).S)}$$

$$\frac{T_1}{T_{min}\,(T_1 + m.S)} = \frac{T_2}{T_{min}\,(T_2 + (1-m).S)}$$

Using equation 23 yields:

$$\frac{(T_1 + m.S)}{NC_1} = \frac{(T_2 + (1-m).S)}{NC_2} \qquad (27)$$

equation 27 tells that when execution time of one cycle of each task is equal (i.e. same frequency for both tasks) the energy consumption will be minimum as shown in figure 5



Figure(5) : Cycle execution time must be equal to all tasks for maximum energy saving

Equation [27] can br rewritten as shown:

$$\frac{T_1}{NC_1} + \frac{m.S}{NC_1} = \frac{T_2}{NC_2} + \frac{(1-m).S}{NC_2} \qquad (28)$$

and from equation [23],

$$\frac{T_1}{NC_1} = \frac{T_2}{NC_2} = T_{min}$$

,then

$$\frac{m.S}{NC_1} = \frac{(1-m).S}{NC_2} \qquad (29)$$

,m can be found from equation[29] which satisfy minimum energy consumption condition

$$m = \frac{NC_1}{NC_1 + NC_2} \,, \quad and \;\; (1-m) = \frac{NC_2}{NC_1 + NC_2} \qquad (30)$$

From last equation ,it can noticed that the optimal value of m of a slack given for each task is independent of amount of available slack(equation[30] is independent of S),rather m is totally dependent on number of execution cycles of each task .Another thing to be noticed is that the task with large number of execution cycles gets most of the slack(i.e. the task with long computation time should have most of the slack).In general ,the optimal slack share for a task for N tasks is given by:

$$Share(task_i) = \frac{NC_i}{\sum_{k=1}^{N} NC_k} * Slack \qquad (31)$$

So the new execution time for $task_i$ would be:

$$ET(task_i)_{new} = T(task_i)_{old} + \frac{NC_i}{\sum_{k=1}^{N} NC_k} * Slack \qquad (32)$$

Equation[32] will be used as task execution time updater .The above equation should take into consideration deadline violation ,and checking policy.

4.4 The optimum energy saving algorithm

For any set of tasks to be optimally scheduled ,it is assumed that tasks are scheduled at maximum processor speed(otherwise it is not scheduled at all).It will be considered that the tasks at V_{max} as a reference point and any subsequent calculations will be based on this point .This is the basis of algorithm-1 given here:

Algorithm 1:

1- **Set the voltage supply = V_{max}**
2- **if ($\frac{\sum_{j=1}^{N} ET_j}{DL_N} \leq 1$ is not true)**
exit(these tasks are not schedulable);
3- **for (i = 1 ; i <= N ; i++)**
{
temp = temp + $ET(task_i)$;
if(temp > $DL(task_i)$)
exit(these tasks are not schedulable);
}
4- **$Slack = DL_N - \sum_{j=1}^{N} ET_j$;**
5- **while(_slack)**
{_slack = 0; temp = 0;
for (j = 1 ; j <= N ; j++)
{ Old_ET = $ET_{old}(task_j)$;
$ET_{new}(task_j) = ET_{old}(task_j) + \left(\frac{NC_j}{\sum_{k=1}^{N} NC_k}\right) * Slack$;
temp = temp + $ET(task_j)$;
if(temp > $DL(task_j)$)
{_slack = _slack + (temp - $DL(task_j)$);}
$ET_{new}(task_j)$= Old_ET;/Retrieve original ET for next run
}//End for
if(_slack > slack)
{
print(The Vmax is optimum schedule);
Run tasks at Vmax;
}//End if
else
slack = slack - _slack;
}// End while
6- **if(slack > threshold)**
7- **for (j = 1 ; j <= N ; j++)**
8- **{ $ET_{new}(task_j) = ET_{old}(task_j) + \left(\frac{NC_j}{\sum_{k=1}^{N} NC_k}\right) * Slack$;}**
9- **$V = f\,[\,ET(task_j)]$;**
10- **V_j = select closet maximum voltage level(V);**
11- **Execute ($task_j$);**
12- **}// End for**
13- **}// End if**
14- **else**
15- **{**
16- **print(The Vmax is optimum schedule);**
17- **Run tasks at Vmax;**
18- **}// End else**

The algorithm first check whether the set of tasks are schedulable or not,for all tasks (step 2) and then for individual task (step 3).Then check the possibility to minimize the energy consumption of tasks execution by stretching the tasks execution times after determining the amount of available slack(step 4) ,updating the tasks execution times .Step 5 will check for possible deadline violation and redistribute the available slack among tasks .Step 6 will check if the slack is larger than threshold and the distribution is worthy and will give significant energy saving.

4.5 Demonstrating the algorithm

Assuming the set of tasks shown which has been generated by a simulater (table-1)

Task	NC. (Mega Cycle)	Old ET.(µSec)	DL.(µSec)
A	26	16	174
B	32	19	211
C	11	6	307

The optimum scheduling according to algorithm-1 is given in

Task	New ET (µSec)	DVS Voltage (V)	Cumulative ET (µSec)	DL. (µSec)	DL-(Cumulative ET) (µSec)
A	116	0.5576	116	174	58
B	142	0.5530	***258***	***211***	***-47***
C	48	0.5574	306	307	1

The percentage of energy saving for above scheduling is 92.35%,but there is a problem in this scheduling, which is that task B has missed its deadline by 47µs (table 2).To solve this problem ,the following procedure should be

1.algorithm should remove the extra time (47µs) from total slack'

2.repeat step for distribution of optimal slack,

3.check for deadline violation

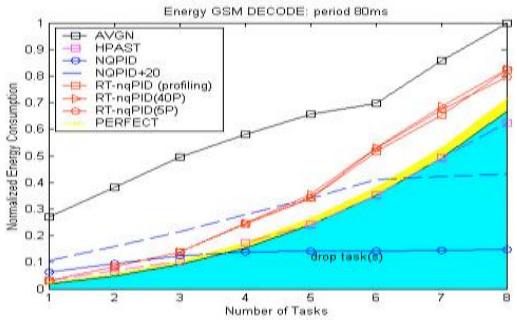
4.checkj the new optimum scheduling.

The algorithm will repeat mentioned procedure above three times to achieve the optimum scheduling shown in table[3]

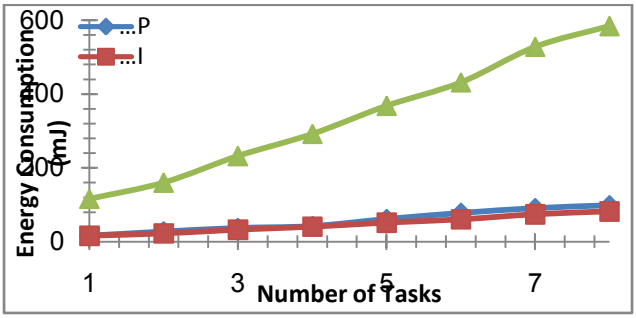
Table-3

Task	New_ET. (µSec)	DVS Voltage (V)	Cumulative_ET. (µSec)	DL. (µSec)	DL-(Cumulative_ET) (µSec)
A	95	0.6079	95	174	79
B	116	0.6103	211	211	0
C	39	0.6172	250	307	57

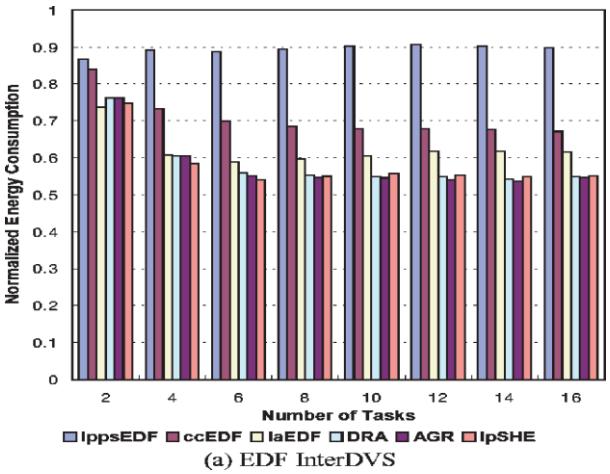
The energy saving achieved now is 90.68% or just 1.8% less than last scheduling ,the program has eliminated 55µs from total slack ,but such elimination makes task scheduling possible and gives a high percentage of energy saving .Figure[6] shows energy consumption for GSM decode for 80 ms-period task, while figure[7] gives energy consumption variations with number of tasks .Figure[8] indicates normalized energy consumption versus number of tasks.



Figure(6) : Energy consumption for GSM decode



Figure(7) : Graphical representation for different tasks execution



Figure(8) : Energy saving achievements for different versions of EDF policy

5. COCLUSION

This paper has studied the individual behavior of each parameter of equation[1],indicating that minimizing a do not always lead to minimum gate cycle time ,It was found that the parameter $(V_j - V_{th})^a$ is very important here ,if its value is nearly 1,then a has to be increased to minimize gate cycle time ,while if its value is less than 1,a has to be decreased .Another point is that minimizing V_{th} would minimize gate cycle time ,but this will maximize static power dissipation ,so there is an optimal design for this value .The optimal slack distribution among tasks was studied and it was found that maximum saving could be achieved if all clock cycles of all tasks where executed with one voltage level(one clock frequency).It was shown that the share of each task from the slack never depend on amount of available slack ,rather it depends on number of execution cycles of each task .The problem of deadline violation was solved also by eliminating the amount of violation from the total available slack.

6. REFERENCES

[1] R.B.Prathipati, 2004.Energy Efficient Scheduling Techniques For Real-Time Embedded Systems .M.Sc .Thesis, Texas A&M University.

[2] S.H,.Gunther,F.Binns,D.M.Carmean,J.C.Hall,2001.Managing the Impact of Increasing Microprocessor Power Consumption .Desktop Platforms Group ,Intel Technology Journal Q1,Intel Corp.

[3] P.Chaparro,G.Magklis,J.Gonzalez,A.Gonzalez,2002.Using MCD-DVS for Dynamic Thermal Management Performance Improvement .Intel Barcelona Research Center ,Intel Labs.

[4] C.h.Hsu, W.C.Feng,2004.Effective Dynamic Voltage Scaling through CPU-Boundedness Detection ,The 4th workshop on power aware Computer Systems ,Portland ,Oregon .Los Almos National Lbs.

[5] J.R.Lorch.2001.Operating Systems Techniques for Reducing Processor Energy Consumption .PhD .Thesis ,University of California ,Berkeley.

[6] K.Nose,2001.Circuit Design for Low-power High Speed VLSI Processor in 0.5 V Generation ,PhD Thesis ,University of Tokyo,

[7] M.Horoitz, T.Indermaur, R.Gonzalez, 2001.Low-Power Digital Design, Center for Integrated Systems, Stanford University.

[8] J.A.Mouw, K.Langendoen, J.Pouwelse, 2002.LART Lessons Learned; cpu frequency, Ottawa Linux Sympoium.

[9] D.Zhu,R.Melhem,B.R.Childers,2003.Scheduling with Dynamic Voltage/Speed Adjustment using Slack Reclamation in Multiprocessor Real-time Systems .IEEE transactions on parallel and distributed systems ,Vol 14,No7.