Energy Constrained - Reliability Aware Scheduling For Real Time System

S.Saroja , S.Muthukumar, N.S.Harini, M.Priyadharshini, N.Ramya

Abstract: This paper solves energy constrained-reliability aware scheduling for real time system. The proposed algorithm considers two main objective functions: energy consumption and system reliability. The primary goal of this project is to minimize the energy consumption and improves the system reliability. Our proposed system is the extension of Earliest Deadline First / Dynamic Deadline Modification policy with static low power scheduling algorithm with shared resources (SLPSR) for periodic task which ignores system reliability. With the help of SPF and LPF algorithms, focused on dynamic low power scheduling algorithm with shared resource for periodic task which uses dynamic slack time which preserves reliability simultaneously saving energy is called DLPSR. Finally, we implemented the proposed algorithm (PDLPSR algorithm) by including the priority of the tasks in to the scheduling decisions. Result shows that the proposed algorithm slightly increases the energy consumption while improving the fault tolerance and system reliability.

Keywords: Reliability; Energy Consumption; Realtime system

1. INTRODUCTION

Energy management plays a vital role in embedded real-time systems. Consuming lower energy can reduce the system expenses and increases the lifetime of the battery. Our objective is to consider reliability and energy consumption at same time by meeting all deadlines. A very simple technique to decrease energy consumption is Dynamic voltage scaling which adjusts the speed of execution by using the execution time. But it has some drawbacks like not considering energy consumption. This is followed in SLPSR algorithm which considers energy consumption but ignores reliability. Then, two heuristic algorithms LPF and SPF which consider reliability but those are based on static slack time. To overcome this, DLPSR algorithm is used, which consider dynamic slack time. Finally, we implemented the proposed algorithm (PDLPSR algorithm) by considering priority. This paper considers a periodic task set containing of 'n' number of tasks with or without a resource requirement. We focused on the uni-processor real time system. Our system consists of a resource set R and periodic task set T. The periodic task set contains 'n' number of periodic tasks, represented as T={T1,T2,…Tn} and the resource set R ={R1,R2,…Rm} has 'm' reusable resources. The resource is a software object (for example: data structure). A periodic task Ti consists of three parameters namely worst case execution time (weti) of Ti, the resource requirement(rri) of the task Ti ,the period(pi) of Ti. Some assumptions made in the paper are,

- *S.Saroja , S.Muthukumar, N.S.Harini, M.Priyadharshini,e N.Ramya Assistant Professor,eResearch Scholar Department of IT*
- *Mepco Schlenk Engineering College,Sivakasi ,Tamilnadu 626005*

- All periodic tasks arrive at zeroth time.
- All tasks are executed in ascending order according to period.

2. RELATED WORK

There are many techniques and algorithms that considers on the energy management while every periodic tasks uses different real-time task models for meeting their deadlines. But all those algorithms took the energy consumption into account and ignored the system reliability. In order to improve the system reliability, checkpoint technique is used but it ignores the processor speed which in turn affects the system reliability. Extending the DVS technique, RA-PM framework is developed which minimizes the energy consumption, also preserves the system reliability. Extending EDF/DDM policy, we proposed an algorithm that focuses both reliability and energy consumption. Reliability is calculated based on the probability of failure.

3. ABBREVIATIONS AND ACRONYMS

EDF - Earliest Deadline first
DVS – Dynamic Voltage Scaling
RA-PM -Reliability-Aware Power Management

4. EDF/DDM Policy

Let Ti, j be the j^{th} instance of the task Ti and rti, jbe the release time of the task Ti. The utilization of the task Ti is given by ui = weti/pi. The total system utilization,Utot=u1+u2+...un. Shortest period is given by Pi = min{pj|rrj=i} where j ranges from 1 to n. According to EDF/DDM policy, for each task instance Ti, j has two kinds of deadlines The initial deadline IDi, j is equal to rti,j+ pi and the execution deadline EDi, j is equal to min{rti,j + pi,ti,j + 1 + Pi} [8]. The power is notated as P.

$$P = P_s + h(P_{ind} + P_{dep}) = P_s + h(P_{ind} + C_{ef} S^m) \quad (1)$$

The critical speed [3] Scrit = 0.3. Ps is static power which is consumed in idle time. It takes the value '0' whenever the system is in off state. Speed-independent power is represented by Pind and will be '0' when the processor is in

idle status. Speed-dependent power is denoted by Pdep = Ce f · S^m where Cef stands for effective switching capacitance. S stands for the running speed of the processor. 'm' stands for system dependent constants (2 ≤ m ≤ 3). Constant coefficient 'h' will be zero when the processor is in idle state or else h = 1.

The reliability of a task means the probability of the task which completes its execution successfully. The reliability of a task Ti under the running speed Si is,

$$R_i(s_i) = e^{-\lambda(s_i)*\frac{wet_i}{s_i}} \quad (2)$$

The Ti (R^0i) is the original reliability of a task which is defined as task executes with the Smax (maximum processor speed) and is expressed as R^0i=Ri (Smax)= $e^{-\lambda 0 \cdot weti}$.

$$\lambda(S) = \lambda_0 g(S) \quad (3)$$

The g(S) is the exponential fault rate model which is expressed as ,

$$g(s) = 10^{\frac{d(1-s)}{1-s_{\min}}} \quad (4)$$

where λ0 is the average fault rate under the Smax (maximum processor speed) , g(Smax) = 1. The minimum processor speed Smin is equal to 0.15 and processor's running speed is denoted as S.

From reliability-aware power management (RA-PM the reliability of the task Ti[7] is given by,
(5)

$$R_i(S_i) = e^{-\lambda(s_i)*\frac{wet_i}{s_i}} + \left(1 - e^{-\lambda(s_i)*\frac{wet_i}{s_i}}\right).R_i^0 > R_i^0$$

Consider the periodic tasks set consisting of three tasks T1, T2, T3 and T1(1,1,4), T2(1,0,5), T3(1.5,1.10).The task T1 and T3 share a same resource R1. The system utilization Utot=u1+u2+u3
So,u1=wet1/p1=1/4=0.25,u2=wet2/p2=1/5=0.2,u3=wet3/p3 =1.5/10=0.15. Utot=0.25+0.2+0.15=0.6. The shortest period of the task Pi is 4 (i.e minimum period of the task set) According to the assumption every task will releases at time t=0, the scheduling interval is between 0 and Z .At time t=0, the task T1 starts execution. It executes until t=1 as the worst case execution time of task T1 is. At time t=1, the task T2 starts execution. It executes until t=2 as the worst case execution of task T2 is 1. At time t=2, the task T3 starts execution. It executes until t=3.5 as the worst case execution of task T3 is 1.5. The task T1 should execute after the period 4 so at time t=4, the task T1 starts execution. It executes till t=5. Likewise, the periodic tasks executes for each period. At time t=12, the task T1 can able to start execute but the another task T3 is executing, so it starts executing from time t=12.5. The completed scheduling using EDF/DDM policy is shown in the figure 1.

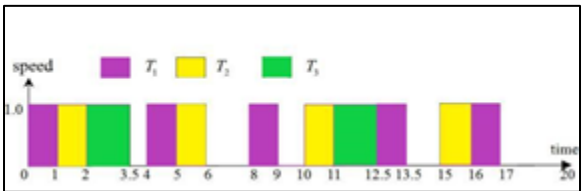

Fig. 1: *Scheduling using EDF/DDM policy*

LPSR algorithm follows the Earliest Deadline First / Dynamic Deadline Modification policy. All tasks should be able to execute successfully without missing any deadlines. Tasks which require resources are grouped as resource requirement task set (RRTS). Tasks which doesn't require resources are grouped as non resource requirement task set (NRRTS)[3,10]. SRRTS(i) is the minimal executing speed of the RRTS and it lies between the interval [0, Z] and deadlines are met by every tasks.. SNRRTS is the minimal executing speed of the NRRTS and it lies between the interval [0, Z] and deadlines are met by every tasks . The periodic task T contains RRTS and NRRTS. SNRRTS is expressed as,

$$S_{NRRTS} = \sum_{T_i \in NRRTS} u_i \quad (6)$$

Where 'ui' as the task utilization and it is expressed as ui = weti/pi .

Every task Ti in RRTS can be executed successfully while meeting the deadline [15]. It should meet the following conditions:

$$Z \geq wet_i + \sum_{j=1}^{i-1} \left\lfloor \frac{Z-1}{P_i} \right\rfloor .wet_j . p_{r_i} < Z < p_i \quad (7)$$

SRT(i) is expressed as follows:

$$S_{RRTS}(i) = \max_{P_{r_i} < Z < p_i} \left\{ \frac{wet_i + \sum_{j=1}^{i-1} \left\lfloor \frac{Z-1}{p_i} \right\rfloor .wet_j}{Z} \right\} \quad (8)$$

Let ES_{RRTS} be the executing speed of RRTS. Thus ES_{RRTS} can be expressed,

$$LS_{RRTS} = \max_{r_i \neq 0 \wedge 1 \leq i \leq n} \{S_{RRTS(i)}\} \quad (9)$$

S_T is the minimal executing speed of the periodic task set T and is expressed as,

$$S_T = LS_{RRTS} + S_{NRRTS} \quad (10)$$

The worst case execution WCET for each task Ti can expressed by $wet_i=(x_i/S_i)+y_i$

Where x_i is the frequency dependent and y_i is the frequency independent .Consider the value of $x_i$and y_i according to the accuracy achieved. For example consider the worst case execution time (WCET) as 1, then x_i and y_i value can be 0.999 and 0.001 respectively to achieve accuracy.

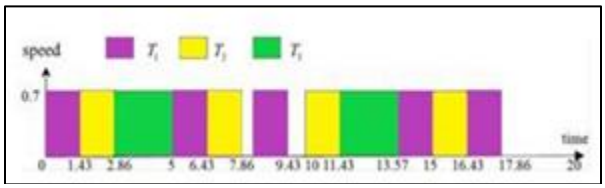

Fig.2: *Scheduling using SLPSR Algorithm*

According to the example considered, the periodic task set (T) is divided as RRTS={T1,T3} and NRRTS ={T2}. The periodic task set (T) is parted as RRTS = {T1, T3} and NRRTS = {T2}. We know u2=1/5=0.2 The SNRRTS for non resource requirement tasks is calculated by the following:

$$S_{NRRTS} = \sum_{T_i \in NRRTS} u_i \qquad \text{(from 6)}$$

$$S_{RRTS}(3) = \max_{4 < Z < 10} \left\{ \frac{wet_i + \sum_{j=1}^{i-1} \left\lfloor \frac{Z-1}{p_i} \right\rfloor . wet_j}{Z} \right\} \qquad \text{(from 8)}$$

So the ST=0.5+0.2=0.7(from 10)

The completed scheduling using SLPSR policy is shown in the figure 2. SPF and LPF algorithm deals with static available slack time. In these algorithms, we choose a scaled task based on its period. In SPF, the shortest period task is chosen as a scaled task. In LPF, the largest period task is chosen as a scaled task. SST=(1-ST)*(ED-tb) where ED is the execution time and tb is the release time of the task. The slack time is used for constructing recovery task. The ST = 0.7 from SLPSR algorithm. At time 2, for task3 ED = 7 and tb = 0 so SST = (1-0.7)*(7-0) = 2.1. The recovery task takes 1.5 slack time. So the speed =1.5/(1.5+2.1-1.5)=0.71. At time 11 ED = 16 and tb = 10, so SST = (1-0.7)*(16-10) = 1.8 Thus Speed = 1.5/(1.5+0.3) =0.83. The same calculation is carried for SPF and LPF it is shown in figure 3 and 4 respectively. In DLPSR algorithm, we consider dynamic slack time instead of dynamic slack time. The actual execution of the task T1,T2,T3 are assumed as 0.5, 0.5, 0.75. Task T1 is assumed as a highest priority task. At T1,1, Speed = 1/(1+1.2-1)= 0.83 and its completes at 0.6 . DST (Dynamic slack time) equals to 0.6+1 = 1.6. At time 0.6, T2,1 has Speed = 1/(1+1.6-1) = 0.63 and it completes at time 1.4. DST = 0.8+1 = 1.8. At time 1.4, the task T3,1 executes with the Speed =1.5/(1.5+1.8-1.5) = 0.83 and ends at time 2.3. DST =0.9+1.5 = 2.4. At time 4, DST = 2.4+1.2=(4-2.3)=1.9. T1,2 has the Speed =1/(1+1.9-1) = 0.53 and ends at time 4.95.DST = 0.95+1 = 1.95. At time 5 DST = 1.95-(5-4.95) = 1.9. T2,2 executes with the Speed =1/(1+1.9-1) = 0.53 and completes at 5.95. The same steps are repeated for other task instances and it is given in figure 5.

STEPS:

1. Calculate the ST from SLPSR algorithm.
2. Calculate the SST for highest priority task.
3. Calculate the Speed and its completion time for every task instance.
4. DST = DST+SST for the first instance of all tasks.
5. For all the other instances,
6. If task == highest priority task, DST=DST+SST-(tb-c) where c is the completion time of previous task.

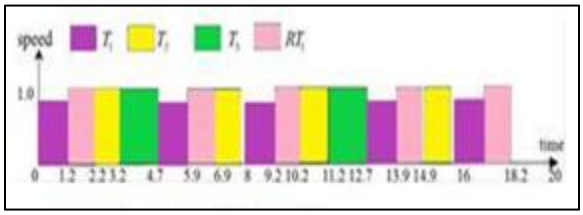

Fig.3: *Scheduling using SPF Algorithm*

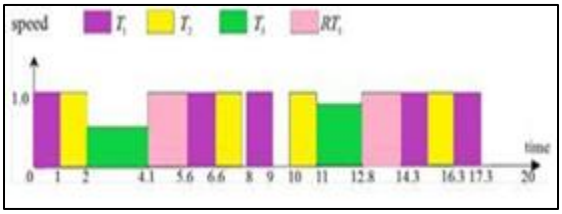

Fig.4*: Scheduling using LPF Algorithm*

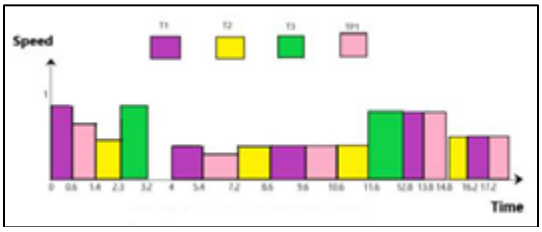

Fig.5: *Scheduling using DLPSR Algorithm*

5. PROPOSED SYSTEM

In Priority Based DLPSR system, we consider dynamic slack time instead of static slack time. The actual execution of the task T1,T2,T3 are assumed as 0.5, 0.5, 0.75, respectively. Task T1 is assumed as a highest priority task based on its period. At T1,1, Speed = 1/(1+1.2-1)= 0.83 and its completes at 0.6 . DST (Dynamic slack time) equals to 0.6+1 = 1.6. At time 0.6, the task with higher priority will execute with Speed=1/(1+1.6+1)=0.63 and it completes at 1.4 . DST(Dynamic slack time) equals to 1.4+1=2.4. At time T2,1 has Speed =0.55 and it completes at time 2.3. DST = 0.8+1 =1.8. At time 2.3, the task T3,1 executes with the Speed = 0.78 and ends at time 3.2.DST = 0.9+1.5 = 2.4. At time 4, T1,2 has the Speed =1/(1+1.9-1) = 0.53 and ends at time 5.4. DST = 0.95+1 = 1.95.The same steps are repeated for other task instances.

6. SIMULATION AND RESULTS

The energy consumption of EDF/DDM policy is (0.08+1.52*1*1*1)*12+0.085*8 = 19.88.The energy consumption of proposed system is (0.08 + 1.52*0.7*0.7* 0.7)* 17.15 + 2.85* 0.085 = 10.56. So, the energy consumption is 46.88% reduced when compared to EDF/DDM policy and it is given in figure 6. The reliability of task also gets increased in the proposed system and the results are given in figure 7.

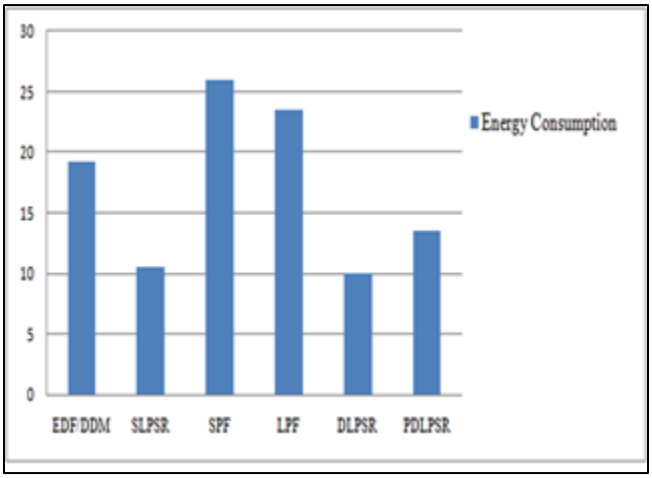

***Fig.6**: Comparative Analysis for Energy Consumption*

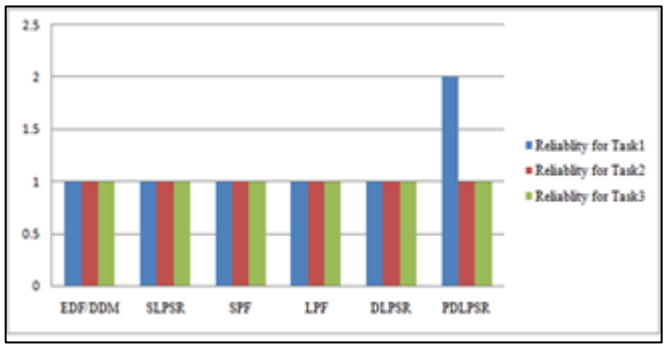

***Fig.7:** Comparative Analysis for Reliability*

7. CONCLUSION

We consider the problem of reducing the energy consumption along with improving the system reliability. First the EDF/DDM policy schedules the periodic task which shares the resources with lowering the energy consumption but not considering the reliability. The proposed algorithm considers the reliability of the system along with minimizing energy consumption by means of incorporating priority and dynamic slack time into account. The simulation result shows that the proposed priority based DLPSR algorithm is better in terms of both energy consumption and reliability.

8 REFERENCES

[1] Yi-wen Zhang, Hui-zhen Zhang, Cheng Wang, Reliability-aware low energy scheduling in real time systems with shared resources Elsevier Microprocessors and Microsystems Volume 52, (2017),Pages 312-324.

[2] Y.-W. Zhang, R.-F.Guo, Power-aware scheduling algorithms for sporadic tasks in real-time systems, J. Syst. Softw. 86 (2013) 2611–2619.

[3] Y.-W. Zhang, R.-F. Guo, Low power scheduling algorithms for sporadic task with shared resources in hard real-time systems, Comput. J. 58 (7) (2015) 1585–1597.

[4] Y.-W. Zhang, R.-F. Guo, Power-aware fixed priority scheduling for sporadic tasks in hard real-time systems, J. Syst. Softw. 90 (2014) 128–137.

[5] D. Zhu, R. Melhem, D. Mosse,´ The effects of energy management on reliabil-ity in real-time embedded systems, in: Proc of IEEE/ACM Int'l Conf. Computer Aided Design (ICCAD'04), San Jose, CA, USA, 2004, pp. 35–40.

[6] D. Zhu, H. Aydin, Energy management for real-time embedded systems with reliability requirements, in: Proc of ICCAD '06, 2006, pp. 528– 534.

[7] D.Zhu,Reliability-aware dynamic energy management in dependable embed-ded real-time systems, in: Proc of 13th IEEE Int'l Conf Real-Time and Embed-ded Technology and Applications Symp, San Jose, CA, USA, 2006, pp. 397–497.

[8] B. Zhao, H. Aydin, D. Zhu, Enhanced reliability-aware power management through shared recovery technique, in: Proc of IEEE Int'l Conf Computer-Aided Design - Digest of Technical Papers, 2009, pp. 63–70.

[9] L.Sha, R. Rajkumar, J. Lehoczky, Priority

[10] inheritance protocols: an approach to real-time synchronisation, IEEE Trans. Comput. 39 (9) (1990) 1175–1185.

[11] K. Jeffay, Scheduling sporadic tasks with shared resources in hard-real-time systems, in: Proc. Real-Time Systems Symp, Phoenix AZ, USA, 1992, pp. 89–99

[12] M.-F. Horng, C.-S. Huang, Y.-H.Kuo, J.-W. Hu, Scheduling sporadic, hard re- al-time tasks with resources, in: Proc of 3rd Int'l Conf Innovative

[13] Computing Information and Control, 2008, pp. 84–87.

[14] X. Castillo, S. X McConnel, D. Siewiorek, Derivation and calibration of a transient error reliability model, IEEE Trans. Comput. 31 (7) (1982) 658–671.

[15] R.K. Iyer, D.J. Rossetti, A measurement-based model for workload dependence of cpu errors, IEEE Trans. Comput 33 (1984) 518–528.

[16] J.-J. Chen, T.-W. Kuo, Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems, in: IEEE/ACM International Conference on Computer-Aided Design, San Jose, USA, 2007, pp. 289–294.

[17] S.Demers Yao, A. Shenker, Scheduling model for reduced CPU energy, in: Proceedings of 36th IEEE Annual Foundations of Computer Science, Milwaukee, WI, USA, 1995, pp. 374–382.