DPVFS: a dynamic procrastination cum DVFS scheduler for multi-core hard real-time systems

Shubhangi K. Gawali* and Biju K. Raveendran

Department of Computer Science,
BITS PILANI University,
Goa, India
Email: shubhangi@goa.bits-pilani.ac.in
Email: biju@goa.bits-pilani.ac.in
*Corresponding author

Abstract: Optimising energy consumption has become the primary focus of research in recent years. Static and dynamic energy optimisation during task scheduling is one of the most prominent measures available. This is achieved mainly by shutdown and slowdown techniques. In uni-processor real-time systems, the most widely used shutdown and slowdown techniques are dynamic procrastination (DP) and dynamic voltage and frequency scaling (DVFS). This paper proposes DPVFS a hard real-time task scheduler for multi-core (MC) system to optimise overall energy consumption without deadline misses. DPVFS combines DP and DVFS for MC systems to save overall energy consumption. DPVFS shuts the processor down whenever possible with the help of procrastination. If shutdown is not possible, it adjusts the voltage and frequency to reduce dynamic energy consumption. The experimental evaluation of DPVFS with synthetically generated benchmark program suites shows savings of 18.8% and 33.2% of overall energy over DP based schedulers and DVFS based schedulers respectively.

Keywords: procrastination; dynamic voltage and frequency scaling; DVFS; multi-core real-time scheduling.

Reference to this paper should be made as follows: Gawali, S.K. and Raveendran, B.K. (2019) ‘DPVFS: a dynamic procrastination cum DVFS scheduler for multi-core hard real-time systems’, *Int. J. Embedded Systems*, Vol. 11, No. 4, pp.461–471.

Biographical notes: Shubhangi K. Gawali is a Lecturer in the Department of Computer Science and Information Systems, BITS PILANI K. K. BIRLA Goa Campus, Goa, India. She received her Bachelor’s in Computer Engineering in 2002 from the Mumbai University and Master’s in 2008 from the NMIMS University. She is currently pursuing her PhD from the BITS PILANI University, BITS PILANI K. K. BIRLA Goa Campus, Goa. Her research interests are in areas of operating systems, real-time systems and database systems.

Biju K. Raveendran is currently serving as an Assistant Professor in the Department of Computer Science and Information Systems, BITS PILANI K. K. BIRLA Goa Campus, Goa, India. He received his PhD from the BITS PILANI, PILANI Campus, Rajasthan in 2009. His research area includes energy efficient multi-core/many-core real-time scheduling, energy efficient memory architecture for multi-core/many-core embedded systems, predictable and dependable real-time/embedded system design, big data systems, etc. He was one of the five recipients of Microsoft Research India Fellowship in 2005 for his PhD work. He is a recipient of Microsoft Young Faculty Award in 2009. He is actively involved in collaborative projects with industries like Microsoft and Aditya Birla Group, etc.

1 Introduction

Energy optimisation has become the primary focus from high performance computing systems to low power embedded systems. While high performance computing systems worry about the heat dissipation, embedded systems worry about the battery life. The major components of energy consumption are dynamic energy due to switching current and static energy due to leakage current. The processors consume static energy in active and idle states. The dynamic energy consumption of a processor in idle state is negligibly small. Both the energy components are negligibly small in shutdown state. Though the switching current reduces with advancement in technology, the leakage current exponentially increases. This makes static energy equally an important component as dynamic energy in overall energy consumption. As shutdown is the only state without static energy consumption, shutting the processor down is the only mechanism to reduce it. This saving is proportional to the length of the shutdown duration as each shutdown has hidden overheads involved with it. Shutting down the core is effective only if the shutdown duration is long. This can be achieved with the help of procrastination techniques (Jejurikar et al., 2004; Lee et al., 2003; Niu and Quan, 2009; Jejurikar and Gupta, 2005; Legout et al., 2013). Procrastination uses the apriori knowledge of the periodic tasks to postpone their execution without missing any timing constraints. Dynamic procrastination can increase the shutdown duration as it uses the run time slack for postponement of job execution. This effectively increases the shutdown duration when executing tasks at maximum voltage and frequency.

Dynamic energy consumption (E_{dyn}) can be reduced by optimising platform independent parameters like number of pre-emptions, cache and memory impacts, etc. Dynamic energy consumption can also be controlled by platform dependent parameters like supply voltage (V) and operating frequency (F) as E_{dyn} of CMOS circuit is given by V^2FC + short-circuit power consumption where C is the load capacitance (Jejurikar et al., 2004). As E_{dyn} has quadratic dependence on supply voltage and frequency, voltage and frequency scaling is the best mechanism to reduce it. Due to run time slack, dynamic voltage and frequency scaling (DVFS) techniques improves dynamic energy saving. The effectiveness of DVFS scheduler depends on the utilisation of idle time and runtime slack.

In a hard-real-time system, quality of the schedule depends only on whether the job is finishing before its deadline. Early execution of a job often results in a non-optimal schedule in terms of energy consumption. The dynamic voltage/frequency scaling and dynamic procrastination are the most widely used techniques to save dynamic energy consumption and static energy consumption respectively. DVFS saves dynamic energy by increasing job execution time. With increase in execution time, the overhead because of pre-emption, cache impacts and decision points increases. This may result in deadline misses as the scaled execution time of the job may become higher than the WCET. DVFS is not very effective when the system utilisation is below a threshold as it is practically impossible to reduce the frequency below a threshold. On the other hand dynamic procrastination is effective when the utilisation is below a threshold and is an overhead at high utilisation. To effectively combine DVFS and dynamic procrastination to achieve the optimal overall energy savings in MC system is still a challenge (Niu and Quan, 2009; Pagani and Chen, 2013; Jejurikar et al., 2004; Gu and Qu, 2013; Khaleel and Zhu, 2016; Hu et al., 2010).

During low processor utilisation, finishing all ready to run jobs at maximum frequency will increase the shutdown duration. Thus static energy consumption depends on task partitioning in use. In statically partitioned cores, for each core, the maximum shutdown time is determined by the task with least period. Shutdown time for most of the cores in the system can be increased if all the tasks with low period are grouped into a set of cores. During high processor utilisation, executing job with reduced voltage and frequency results in dynamic energy saving as well. It is also known that applying DVFS slims down the chances of future shutdown. Combining these approaches will offer optimal energy saving while satisfying the quality of schedule. The paper proposes a dynamic hard real-time priority driven scheduler with 100% schedulability which uses the slack generated by the early finishing of jobs at runtime. The objective of this work is to come up with a MC schedule which has minimum idle time with the help of DVFS and DP.

The rest of the paper is organised as follows: Section 2 discusses the recent literature in the area of processor slowdown and shutdown techniques for MC systems. In Section 3, the proposed energy efficient MC real-time scheduler – DPVFS is explained. Schedulability analysis, correctness and complexity of algorithms are explained in Sections 4, 5 and 6 respectively. Section 7 discusses the experimental setup. The experimental results in comparison with DP and DVFS are presented in Section 8. Section 9 concludes the paper with future directions.

2 Related work

2.1 *Slowdown techniques while scheduling in MC systems*

Pillai and Shin (2001) propose architecture dependent slowdown techniques like static voltage and frequency scaling (SVFS) and DVFS to save dynamic energy consumption during the active period by reducing the supply voltage and clock frequency. The SVFS selects the lowest possible supply voltage and operating frequency at worst case execution time (WCET) that allows earliest deadline first (EDF) and rate monotonic (RM) schedulers to meet all the deadlines for a given periodic task set (Pillai and Shin, 2001). While executing, a job may finish well before its WCET which generates slack. This slack can be utilised by executing the future jobs at reduced voltage and frequency. DVFS uses this slack to reduce the voltage and frequency further. Pillai and Shin (2001) proposed two DVFS schedulers – cycle conserving (CC) and look ahead (LA). In CC, the voltage and frequency are recomputed using the actual execution time (AET) of the finished jobs and the WCET of jobs which are there in the queue. They showed that 20% to 40% of energy saving can be achieved by employing this technique. The LA technique determines the future computation needs and defers the task execution by setting the operating frequency as low as possible by ensuring all future deadlines. Aydin et al. (2004) explained two DVFS schedulers – dynamic reclaiming and aggressive speed reduction. Yang et al. (2005) proposed an approximation algorithm for DVFS scheduling on multiprocessor. Aydin et al. (2001) proposed DRA and AGR algorithms. DRA keeps track of dispatch times of tasks. During runtime if a task is dispatched earlier, the processor is slowed down to prolong the execution time till the original completion time. AGR estimates the task completion time from the past history and computes the lowest processor speed. Nassiffe et al. (2016) proposed similar solution but have assumed that CPU frequency can be selected continuously within a given range. Supply voltage and clock frequency can be optimised only if the architecture supports discrete voltage and frequency levels (Yang et al., 2005; Aydin et al., 2004; Pillai and Shin, 2001). For high end compute intensive applications to optimise make span genetic algorithm (GA) and immune genetic algorithm (IGA) are used in computational grid (Shiv and Deo, 2014; Prakash et al., 2016; Trivedi et al., 2016). These works apply IGA and bat algorithm to schedule the submitted jobs on the grid nodes for the optimal make span. Though various slowdown methods are used to reduce the dynamic energy these methods do not address static energy saving. Rather it is done at the cost of increased execution time which inherently increases static energy. Static energy may also increase if the supply voltage is scaled down beyond threshold. According to Jejurikar et al. (2004) the critical speed for 70 nm technology is 0.4 which means the static energy consumption increases if the supply voltage is scaled beyond 0.4. This enforces limitation on reducing the supply voltage which does not allow minimising the dynamic energy consumption further.

2.2 *Shutdown techniques while scheduling in MC systems*

With advances in fabrication technology, the leakage current and thus static energy increases exponentially because of the reduction in channel length of CMOS transistors (Jejurikar et al., 2004). The shutdown techniques not only save static energy but also save dynamic energy as well (Jejurikar et al., 2004; Devdas and Aydin, 2012; Chen et al., 2017; Liu et al., 2017). Procrastination is one of the most widely used techniques to increase the shutdown duration for saving static and dynamic energy consumption (Jejurikar et al., 2004; Jejurikar and Gupta, 2005; Lee et al., 2003; Niu and Quan, 2009; Legout et al., 2013). Procrastination uses the apriori knowledge of the periodic tasks to postpone their execution without missing any timing constraints. This increases the shutdown duration and thus saves more static energy. Procrastination increases the chance of shutting down the processor otherwise would have been in idle state which helps in saving static and dynamic energies. The leakage current is reported as 0.01 A/m for the 130 nm and 3 A/m for 45 nm technology (Jejurikar et al., 2004). To reduce static energy, various shutdown techniques are proposed when processor(s)/core(s) are not in use (Devdas and Aydin, 2012; R. et al., 2004; Gabriel and De Mello, 2015; Wu, 2014). Various static and dynamic procrastination solutions have been proposed for real-time systems under different application/device settings (Devdas and Aydin, 2012). In static procrastination techniques, the latest start time of every job is pre-computed by considering the WCET and other timing constraints of all the jobs (Devdas and Aydin, 2012; Jejurikar et al., 2004). Most of the jobs completes before WCET thereby leaving some slack before its deadline. The dynamic procrastination technique uses these slacks to increase the shutdown duration of the processor (Lee et al., 2003; Legout et al., 2013; Niu and Quan, 2009). Lee et al. (2003) explained how idle duration can be extended. It can be done by extending the active period with an interval given by the sum of pseudo execution time of all tasks that are pre-empted plus the delay of the currently executing earliest deadline task before the active period started. Niu and Quan (2009) extended the idle interval length by computing the latest start time of the job set without missing deadlines of future jobs. Procrastination is feasible only if the shutdown duration is beyond the threshold otherwise the associated overhead of processor shutdown and wakeup can overkill the saving made by shutting down the processor. In Meisner et al. (2009) explained how PowerNap concept outperform DVFS approach in realistic applications. Chen and Kuo (2006) proposed a method to simulate the execution of periodic tasks to compute the idle time available until the next deadline. This idle time is used for postponing the tasks to shut the system down. They also proposed virtual blocking which is the maximum blocking that tasks can suffer to extend the procrastination interval.

Awan and Petters (2011) proposed accumulation of task execution slack and switch the processor off during such intervals under EDF. Huang et al. (2009) proposed an offline analysis that combines DPM and real-time calculus. This method estimates tasks arrivals, computes CPU idle intervals and modulates between active and sleep states at runtime. In both these methods, the tasks are always executed at the maximum speed.

2.3 *Combining slowdown and shutdown techniques while scheduling in MC systems*

Slowdown or shutdown technique alone does not offer optimal energy saving. To balance both static and dynamic energy consumptions, sometimes the tasks are required to execute at necessary speed and sometimes at higher than necessary speed to increase the idle intervals (Niu and Quan, 2009; Lin et al., 2017). Niu and Quan (2009) proposed DVSLK which finds the latest start time of jobs and merges the scattered idle intervals into larger ones such that no jobs miss the deadline. This algorithm uses static schedule with reduced voltage and frequency for each job defined statically and procrastinates only when no job is ready for execution. Pagani and Chen (2013) adopted a simple and linear-time strategy called single frequency approximation (SFA) on multi-core system. SFA executes all jobs at critical frequency along with the procrastination scheme. Jejurikar et al. (2004) proposed CS-DVS-P based on the critical speed and task procrastination. CS-DVS-P uses offline DPM method to compute the maximum time each task can spend in the sleep state within its period and then at run time the controller switches the system off for the corresponding pre computed duration. Both SFA and CS-DVS-P algorithms give optimal result only when all tasks execute for their worst case.

3 Proposed work

This work proposes dynamic procrastination with voltage and frequency scaling (DPVFS) scheduler an energy efficient scheduler for MC hard real-time systems. It uses DP and DVFS to save static and dynamic energies. DP helps in converting the distributed short idle durations to longer shutdown duration. The unused idle intervals are converted into active by executing the jobs at reduced voltage and frequency using DVFS. In DPVFS, before executing a job, the next idle duration is computed using procrastination method (Jejurikar et al., 2004; Lee et al., 2003; Niu and Quan, 2009; Jejurikar and Gupta, 2005; Legout et al., 2013). If significant length of idle duration is available in future, the current job is executed at maximum voltage and frequency. Otherwise the scaled voltage and frequency for the job is computed using DVFS and the job execution is extended for the available idle duration. DPVFS recommends a choice of slowing down the processor or shutting down the processor based on procrastinated idle duration. To obtain optimal energy consumption, this work systematically incorporates DVFS and DP to balance the dynamic and static energy savings for the architectures that support multiple voltage and frequency levels and processor shutdown. DPVFS uses modified first fit bin packing (MFFBP) (Gawali and Raveendran, 2016) approximation for task allocation and either cycle conserving earliest deadline first (CCEDF) (Pillai and Shin, 2001) or dynamic procrastination scheduler (DPS) for job scheduling (Niu and Quan, 2009). MFFBP arranges tasks in non-decreasing order of their periods to increase the shutdown duration of the processors. In MFFBP, to increase shutdown duration, the tasks with shorter periods are allocated to a set of processors and the processors with shorter period tasks are not considered for shutdown. This allows the processors having higher period tasks to have higher shutdown duration thus better static energy saving. The DPS scheduler offers the maximum shutdown duration possible by taking care of all job deadlines. The shutdown duration and shutdown threshold helps in deciding whether to keep the processor in shutdown state or in idle state. At any point in time, the optimal procrastination time can be computed by considering all the future jobs from the current time till hyperperiod. The exponential computation time complexity makes this impractical. In DPS, the shutdown duration is calculated by considering the jobs arriving between current time t and D_{next} where D_{next} is the deadline of the lowest priority job arriving before the nearest deadline. All the arrivals are considered for their WCET. For the jobs whose deadlines are after D_{next}, only a portion of their executions before D_{next} are considered. The DPVFS algorithm decides whether to slowdown or shutdown the processor based on procrastinated time and shutdown threshold. If the procrastinated shutdown duration computed by DPVFS is more than the shutdown threshold, the processor shuts down for procrastinated shutdown duration computed by DPS algorithm after backing up the relevant data. Otherwise the system follows DVFS schedule by keeping the processor idle until the next job arrival. The DPVFS scheduler assumes that all the N hard real-time tasks are in-phase, independent, pre-emptive and periodic in nature whose deadlines are same as their periods. Each task T_i of a task set T is represented by $\{\phi_i, P_i, C_i, D_i\}$ (Liu, 2004). The kth job of task T_i is represented by $J_{i,k} : \{a_{ik}, c_{ik}, d_{ik}\}$ where $d_{ik} = a_{ik} + D_i$. The utilisation of task T_i is calculated as $U_i = C_i / P_i$ which is the fraction of processor time used by T_i. The total processor utilisation is calculated as $U_p = \sum_{i=1}^{N} U_i$. The hyperperiod over which the task set is scheduled is computed as least common multiple of all P_i.

Abbreviations: wakeup time (WT), next job arrival time (NJAT), active period (AP), idle duration (ID), shutdown threshold (ST), scaled voltage (SV), scaled frequency (SF), next idle duration (NID), running job (R), array (W) of all jobs of S_i ready at t.

The DPS, selector and DVFS algorithms are given below:

```
Algorithm 1: DPS
    Input: time t which is the end of active period,
           sub taskset $S_i$ allocated to the processor $P_i$
    Output: ID from t
1   Initialise $P_i.AP = P_i.ID = 0$ and
    $P_i.SV = MAXVOLTAGE$
2   Find job $J$ with earliest deadline as $D_{next1}$ after $t$
3   if ($D_{next1} - t - J.WCET < SDT$) then
4       return PID = NJAT
5   Find $D_{next2}$ as deadline of lowest priority job
    arriving before $D_{next1}$
6   Find all jobs (L) releasing between $t$ and $D_{next2}$
    sorted in non-increasing order of their absolute
    deadlines and initialise WT to $D_{next2}$
7   foreach job $J_i \in L[]$ do
8       if ($J_i.d > D_{next2}$) then
9           WT –= (($D_{next2} - J_i.r$) * ($J_i$.WCET / $J_i$.P) * $U_p$)
10      else
11          WT –= $J_i$.WCET
12      if ( $J_i$ ! = lastjob & $WT > J_{i+1}.d$) then
13          WT = $J_{i+1}.d$
14      end
15  end
16  return ID = WT – t
```

```
Algorithm 2: SELECTOR
    Input: $\forall_{i=1 \text{ to } P}$ sub taskset $S_i$ []
    Output: Scaled voltage and frequency for
            processor $P_i$ ($P_i$.SV)
1   foreach job $J_i \in W$ [] do
2       $C_i$.AP = $C_i$.AP + $J_i$.WCET
3       W[] ← jobs arriving between $t$ and $C_i$.AP
4   end
5   if ($P_i.AP$ == 0) then
6       $P_i$.SV = MAX VOLTAGE
```

```
7     $P_i$.NID = DPS($P_i$, $t + P_i$.AP)
8     **if** ($P_i$.NID < SDT) **then**
9         $P_i$.SV = nearest higher voltage than
              ($P_i$.AP / ($P_i$.AP + $P_i$.NID))
10    **return** $P_i$.SV
```

Algorithm 3: DPVFS

```
      Input: $\forall_{i=1 \text{ to } P}$ sub taskset $S_i$[]
      Output: Schedule with voltage scaling or procrastination
1     Event1: On Arrival of job J
2     Update R.WCET, $P_i$.AP and $P_i$.NID with R.AET in SF
3     if ($P_i$.AP == 0) then
4         SELECTOR($P_i$)
5         Execute job J at Pi.SV;
6     end
7     else
8         Execute highest priority job at $P_i$.SV
9     End Event1
10    Event2: On Completion of job J
11    Update R.WCET, $P_i$.AP & $P_i$.NID with R.AET in SF
12    if (($P_i$.AP == 0) && ($P_i$.NID == 0)) then
13        SELECTOR($P_i$)
14    if ($P_i$.NID ≥ SDT) then
15        if ($P_i$.RQ is empty) then
16            Keep processor $P_i$ in Shutdown state till
                  $P_i$.NID
17            $P_i$.SV = MAX VOLTAGE
18        end
19        else
20            Execute highest priority job at MAX
                  VOLTAGE
21    end
22    else
23        if ($P_i$.RQ is empty) then
24            Keep processor $P_i$ in Idle state till next job arrival
25            $P_i$.SV = MAX VOLTAGE
26        end
27        else
28            $P_i$.AP = $P_i$.AP * $P_i$.SV
29            $P_i$.SV = nearest higher voltage of ($P_i$.AP – J.AET
              in SF) / ($P_i$.AP – J.AET in SF + $P_i$.NID)
30            $P_i$.AP = $P_i$.AP / $P_i$.SV
31            Execute highest priority job at $P_i$.SV
32        end
33    end
34    End Event2
```

Consider a task set T consisting of seven periodic hard real-time tasks T_0 to T_6 represented as (period, WCET) $T_0(50, 20)$, $T_1(60, 15)$, $T_2(80, 19)$, $T_3(110, 15)$, $T_4(100, 20)$, $T_5(140, 25)$ and $T_6(120, 20)$. According to MFFBP, two processors are required to execute this task set. Basedon period, the tasks are ordered as $\{T_0, T_1, T_2, T_4, T_3, T_6$ and $T_5\}$. The task set is divided into two sub task sets $S_1\{T_0, T_1, T_2\}$ and $S_2\{T_4, T_3, T_6, T_5\}$. S_1 and S_2 are allocated to the processors P_1 and P_2 respectively for scheduling. Consider scheduling of jobs of tasks in S_1 with AET same as estimated WCET. At the beginning of the active period, i.e., at t = 640, the selector function is invoked. The jobs in W are $\{J_{2,8}, J_{0,13}, J_{1,11}\}$ and Step 1 of SELECTOR algorithm finds P_1.AP as 54. Step 2 guarantees the initialisation of scaled voltage value at the end of each active period. Step3 calls DPS function to compute P_1.NID at t = 694 (640 + 54). DPS function in this case returns six as PE_1.NID. Assuming ST as 25, the processor decides to slowdown and the PE_1.SV is calculated as 54 / (54 + 6) = 90% of the maximum voltage and frequency. The jobs $J_{2,8}$, $J_{0,13}$ and $J_{1,11}$ are executed with reduced voltage and frequency between 640 and 700. At t = 700, the next active period begins with only one job $J_{0,14}$ in W. There is no idle time before 700 + $J_{0,14}$.WCET = 720. Thus job $J_{0,14}$ executes with full voltage and frequency till 720. At t = 720, P_1.AP = 54. At t = 774 (720 + 54), DPS computes P_1.NID as 30 by procrastinating the job that arrives at 780. Since P_1.NID ≥ ST, P_1.SV is set to maximum voltage and the tasks are executed with full voltage and frequency from t = 700 till 774. The processor is switched to shutdown state from t = 774 till 804. Thus the jobs in active period before the processor shutdown are executed at maximum voltage and frequency. Using the same method PE_1.SV is computed as maximum voltage till t = 920, 90% of maximum voltage till t = 950 and maximum voltage till t = 1039. The processor remains in idle state till 1,040. The resultant schedule using DPVFS and DPS (Gawali and Raveendran, 2016) till time 1,040 are shown in Figure 1 and Figure 2 respectively.

jobset1	J0, 13, J2, 8, J0, 11
jobset2	J0, 14, j1, 12, J2, 9, J0, 15
jobset3	J0, 16, J2, 10, J1, 13, J1, 14, J0, 17, J2, 11, J0, 18, J1, 15
jobset4	J0, 19, J1, 16, J2, 12, J0, 20, J1, 17

Figure 1 Gantt chart using DPVFS

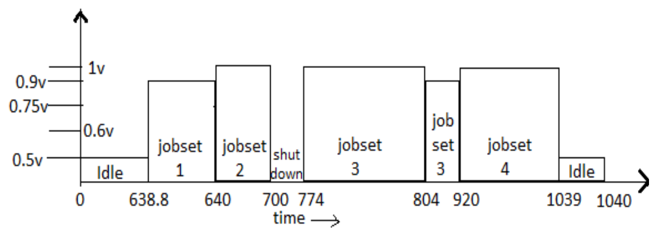
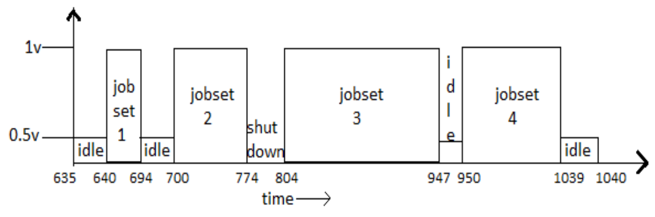


Figure 2 Gantt chart using DPS



4 Schedulability analysis

Theorem 1: Any periodic task set T with implicit deadlines can be feasibly scheduled on m identical cores by DPVFS algorithm on satisfying the following condition.

$$U_{tot} \leq m(1 - U_{\max}) + U_{\max} \tag{1}$$

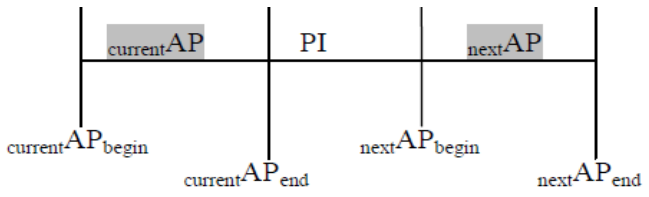
where $U_{tot} = \sum_{i=1}^{N} U_i$ and $U_{\max} = \max(T_i.WCET = T_i.P)$ and $T_i \in T$.

Proof: Since the multi-core system follows static task allocation for a periodic task set, schedulable utilisation of the task set can be found using equation (1). It gives the sufficient schedulability condition for a system containing m identical cores, each scheduled on dynamic priority basis. Since DPVFS is a task level dynamic priority algorithm, it always finds a feasible schedule for T that satisfies equation (1).

Theorem 2: Any periodic task set T with implicit deadlines satisfying equation (1) can be feasibly allocated to at most m identical cores, each executing a dynamic scheduling algorithm, using MFFBP allocation algorithm such that the overall shutdown duration is high.

Proof: MFFBP follows bin packing approximation. All the tasks are assumed to be hard affined without migration. MFFBP sorts tasks based on period and allocates them to cores based on first fit approximation. The smaller period tasks are grouped to a set of processors which makes the remaining set having higher probability of shutdown. Since MFFBP considers task level dynamic priority scheduling algorithms with 100% utilisation in each core, it maintains the schedulability of bin packing approximation.

Corollary 1: The periodic task set T with implicit deadlines remain schedulable after voltage/frequency scaling.



Proof: Let AP denote active period and PI be procrastination interval. The slowdown decision is made at the beginning of the AP ($_{\text{current}}\text{AP}_{\text{begin}}$) and on completion of job if it finishes earlier than its WCET thereby leaving slack before the end of AP ($_{\text{current}}\text{AP}_{\text{end}}$). It is decided to slowdown only if the PI is less than the shutdown threshold. The slack produced by the jobs is utilised to slowdown other jobs till the beginning of next active period ($_{\text{next}}\text{AP}_{\text{begin}}$). Thus it does not affect the schedulability. The frequency of the processor chosen for execution of jobs in $_{\text{current}}\text{AP}$ is:

$$f = \sum_{\forall j_i \in J} j_i.AET + \sum_{\forall j_k \in K} j_k.WCET \leq {}_{\text{current}}\text{AP} + \text{PI} \tag{2}$$

where J is the set of completed jobs before t in current active period and K is the set of incomplete jobs in remaining active period.

Equation (2) shows that no higher priority job in $_{\text{current}}\text{AP}$ will miss its deadline due to extension of lower priority jobs. Thus the task set remains schedulable after voltage and frequency scaling.

Corollary 2: The periodic task set T with implicit deadlines remain schedulable on procrastination.

Proof: The shutdown decision is made at time t when the ready queue is empty, i.e., at the end of AP ($_{\text{current}}\text{AP}_{\text{end}}$). It is decided to shutdown only if the PI is more than the shutdown threshold. Assume that all the jobs in $_{\text{next}}\text{AP}$ are arranged in chronological order such that j_i has higher priority than j_j, j_j has higher priority than j_k and so on. Let J_i be the set of jobs in $_{\text{next}}\text{AP}$ released before D and having deadline before D and ET_i be the total execution time of the jobs in J_i. Let J_l be the set of jobs in $_{\text{next}}\text{AP}$ released before D and having deadline after D. Let ET_l be the total execution time of the jobs in J_l. Thus

$$ET_i = \sum_{\forall j_i \in J_i : i \neq k \,\&\, ri, di \leq D} C_i \tag{3}$$

$$ET_l = \sum_{\forall j_l \in J_l : i \neq k \,\&\, rl \leq D, dl \leq D} D - rl \tag{4}$$

At time t, procrastination duration Z_k of job j_k (job with earliest deadline after t) is computed using the following condition:

$$C_k + Z_k + ET_i + ET_l \leq \text{PI} + {}_{\text{next}}\text{AP} \tag{5}$$

where D is the absolute deadline of job having release time $r_j < d_k$ and deadline $d_j > d_k$, J_i is the set of jobs in $_{\text{next}}\text{AP}$ released before D and having deadline before D, J_l is the set of jobs in $_{\text{next}}\text{AP}$ released before D and having deadline after D.

On procrastination, all the tasks in next active period will be executed with full frequency. Equation (5) shows that no job will miss its deadline due to postponement of job execution and the task set remains schedulable with procrastination.

Theorem 3: Any periodic task set T_{ij} with implicit deadlines having less than or equal to 100% utilisation is DPVFS schedulable on a core j.

Proof: DPVFS is a task level dynamic priority scheduling algorithm for individual core with decision points as job arrivals, departures and wakeup. For a given task set T_{ij} on core j, DPVFS offers a valid schedule if it satisfies the utilisation bound given in equation (6).

$$\sum_{i=1}^{N} U_{ij} \leq 1 \tag{6}$$

where U_{ij} = Utilisation of $T_i \in T$.

The slowdown and procrastination decisions in DPVFS make sure it maintains the schedulability bound as shown Corollary 1 and Corollary 2.

5 Correctness of DPVFS

Theorem 4: DPVFS produces a valid and feasible schedule if there exist one.

Proof: For periodic task set with implicit deadlines having hard affinity, any priority driven scheduler is invoked on job arrival and completion. DPVFS also has these decision points but it has to not only select the highest priority job from ready queue but also has to decide the state of the core (slowdown/shut down). Corollary 1 shows that a job can be feasibly scheduled even on executing for longer duration without any deadline misses. Corollary 2 shows that a job can be feasibly scheduled even after postponing its execution without any deadline miss. Thus DPVFS produce a valid and feasible schedule if there exist one in voltage/frequency scale down and procrastination decision.

Theorem 5: DPVFS produces a schedule with equal or lesser energy consumption compared to EDF, DVFS and DP algorithms.

Proof: Let S_{DVFS}, S_{DP}, S_{EDF} and S_{DPVFS} be the schedules produced by DVFS Pillai and Shin [2001] DP G. and R. [2016], EDF and DPVFS scheduling algorithms. Let SE_{EDF}, SE_{DVFS}, SE_{DP} and SE_{DPVFS} be the static energy consumed and DE_{EDF}, DE_{DVFS}, DP and DE_{DPVFS} be the dynamic energy consumed while scheduling S_{EDF}, S_{DVFS}, S_{DP} and S_{DPVFS} respectively. The trend of static and dynamic energy consumptions of the algorithms is as follows:

$$SE_{DP} \leq SE_{DPVFS} \leq \{SE_{EDF}, SE_{DVFS}\}$$
$$DE_{DVFS} \leq DE_{DPVFS} \leq DE_{DP} \leq DE_{EDF}$$

DVFS algorithm takes care of only dynamic energy optimisation whereas DP algorithm takes care of static energy optimisation. Since DPVFS follows shutdown whenever possible and follows slowdown otherwise, it gets advantage of both static and dynamic energy saving. This results in reduced overall energy in comparison with all other approaches. Thus the overall energy consumption in S_{DPVFS} will be lesser than or equal to S_{EDF}, S_{DVFS} and S_{DP}.

6 Complexity of DPVFS

DPVFS is a task level dynamic priority scheduling algorithm with decision points as:

1 job(s) arrival

2 core wakeup

3 job completion.

When the core is awake, on job(s) arrival, the job(s) joins the ready queue which is maintained as priority queue in $N \log N$ time where N is the number of tasks allocated to the core. Selecting next job to run with same voltage and frequency is a constant (C_1) time operation. Thus the run time complexity of job scheduling on job(s) arrival when core is awake is $O(N \log N)$.

If the job(s) arrives when the ready queue is empty, the active period and next idle duration is computed after the job insertion. The active period is computed by considering WCET of all the jobs which arrives before the end of active period. It is seen experimentally that the number of jobs varies from 1 to $2 * N$. Thus it takes $O(2N)$ time to compute active period. The idle duration from the end of active period is computed by considering the WCET of jobs between earliest release time of job after the end of current active period and deadline of the lowest priority job arriving before the nearest deadline (D_{next}). It takes $O(L\log L)$ for finding and arranging the jobs based on their deadline where L is the number of jobs arriving between end of active period and D_{next}. Thus it takes $O(L\log L)$ time to compute next idle duration. Thus the run time complexity of job scheduling when ready queue is empty is $O(N \log N + L\log L)$.

Though the transition from shutdown to active is time consuming, for complexity calculation it is considered as constant (C_2) time operation. When the core wakes up from shutdown, all the pending jobs joins the priority ready queue in $N \log N$ times. Selecting next job to run with maximum voltage and frequency is a constant (C_1) time operation. Thus the run time complexity of job scheduling on core wake up is $O(N \log_N)$.

On job completion, the active and idle periods are updated in constant (C_3) time. Then a decision to slowdown or procrastinate is taken which takes constant time (C_4). Accordingly the core is transited to shutdown or idle state and appropriate voltage level setting inconstant (C_2) time.

If the ready queue is non-empty and the decision is shutting down the core when it becomes idle, the voltage level is set to MAX VOLTAGE. If the decision is to slow down, the appropriate voltage and frequency is computed. Thus like other seminal dynamic priority algorithms, DPVFS also takes constant (C_5) time for scheduling a job on job completion. Thus the run time complexity of DPVFS on job completion is $O(1)$.

$$\text{MAX}\{O(N\log N + L\log L), O(N\log N), O(1)\}$$
$$= O(N\log N + L\log L).$$

7 Experimental setup

A new framework named simulator for multi-core real-time scheduler (SMCRTS) is designed and implemented to find schedule and measure energy parameters like idle, static, dynamic and total energy consumptions of DPVFS, CCEDF, and DP schedulers. Similar to Jejurikar et al. (2004), Lee et al. (2003), Niu and Quan (2009), Legout et al. (2013), Renaux (2014) and Shajulin et al. (2017), the experimentation is conducted using several randomly generated task sets each containing 20 tasks. Such randomly generated tasks are used as the common validation methodology in real-time scheduling. Based on Jejurikar et al. (2004), tasks were assigned a random period between the range [250 ms, 8,000 ms] and WCET between 35% and 80% of the period such that the total utilisation of the task set varies from 265% to 305% in a set of four cores. The experiments obtain the AET using Gaussian distribution with Rmean, μ = (WCET + AET) / 2 and standard deviation $= \int (\text{WCET} - \text{AET})/$ number of tasks where WCET denotes the WCET. The AET of the task is varied between 10% and 100% of its WCET in steps of 10%. The experimentation is conducted by using ten discrete voltage levels common to all the cores ranging between minimum 0v to maximum 1v with step of 10% increment for different states of core. The power consumption of the core is computed with respect to minimum voltage when it is in shutdown state, 50% of maximum voltage when it is in idle state and 50% to 100% when it is in active state. All tasks are simulated to execute till hyperperiod, i.e., least common multiple of periods of all tasks in the task set since the pattern of the schedule repeats after hyperperiod. All these algorithms follow first fit decreasing arranged based on their periods.

The energy components are considered for 70 nm technology for Transmeta Crusoe processor reported by Jejurikar et al. (2004). The total energy consumption (E_{tot}) during execution is measured by considering energy components like static energy (E_{stat}), dynamic energy (E_{dyn}), processor shutdown and wakeup energy (E_{PSD}), scheduler decision making energy (E_{sched}) and idle state energy (E_{idle}) while executing with various scheduling algorithms. E_{stat} is the energy due to leakage current and reverse bias junction current. It is computed as product of active duration and static energy per unit. The static energy is assumed to be 22nJ per cycle. E_{dyn} is the energy due to switching activity in a circuit. It is computed as V^2fC where V is supply voltage, f is clock frequency and C is load capacitance. According to Jejurikar et al. (2004), the maximum frequency at 1 volt is 3.1 GHz with 0.43 nF of capacitance. Based on Jejurikar et al. (2004), the idle state dynamic energy is considered to be 11 nJ per cycle. Since the experiments are performed on task sets at various voltage levels and frequencies, the dynamic energy is considered in the range 11 nJ to 44 nJ per cycle. E_{PSD} is the energy due to flushing of data cache during shutdown and memory accesses during wakeup. It is computed as product of number of shutdown decisions and shutdown overhead. The shutdown overhead is estimated by considering the on chip cache and other storage infrastructures. The cache size of the processor is assumed to be 32 KB I-cache and 32 KB D-cache. It is assumed that 20% lines of data cache are dirty before shutdown which results in 6,554 memory writes. By considering 13 nJ of energy per memory write, the total energy for data cache flush is 85 μJ. The energy and latency of saving the registers is assumed to be negligibly small. When a task resumes its execution, the locality of reference changes which causes cache misses. This additional cache misses is assumed to be 10% of the cache size in both I-cache and D-cache. This causes the total overhead of 6,554 caches misses on wakeup. By considering 15 nJ of energy per memory access, the total energy required for reading from memory and writing into cache is 98 μJ. The energy required for updating TLBs and BTBs is assumed to be negligibly small. The energy required for charging the circuit logic is assumed to be 300 μJ. Thus the total energy required switching the processor between active and shutdown state is 85 + 98 + 300 = 483 μJ. E_{sched} is the energy consumed for decision making at every scheduler invocation. It involves cost of computing procrastinated idle duration whenever the processor is idle or cost of selecting the highest priority job from ready queue and allocating processor to it whenever some job is ready to execute. It also involves context switching overhead caused due to task completion or pre-emption and cache impact overhead caused due to context switching. E_{sched} is computed as the product of total number of scheduler invocations and decision making energy for scheduling or procrastination. The scheduling decision energy is assumed to be 2 μJ for voltage and frequency selection, 2 μJ for procrastination decisions and 40 μJ for job dispatcher. E_{Idle} is the energy consumed during idle period. It is computed as the product of idle duration and energy per unit time. According to R. et al. [2004], the idle state energy is assumed to be 33 nJ for shutdown interval threshold of 2 ms. Total energy consumption.

$$E_{tot} = E_{stat} + E_{dyn} + E_{PSD} + E_{sched} + E_{Idle} \tag{7}$$

8 Experimental result

Figure 3 shows the static energy consumption per unit time for different utilisations. DPS consumes the least static energy followed by DPVFS and CCEDF. This is because the static energy consumption is inversely proportional to the shutdown duration. The shutdown duration is more in DPS algorithm compared to DPVFS and CCEDF. This is because DVFS reduces the chance of shutdown as it increases the execution time of jobs. The DPVFS algorithm offers very close performance in comparison with pure procrastination based algorithms which show its conversion accuracy from idle to shutdown. On an average DPVFS reduces the static energy by 84.54% over CCEDF and has 1.45% more static energy consumption over DPS.

Figure 3 Static energy consumption per unit time for different utilisations (see online version for colours)

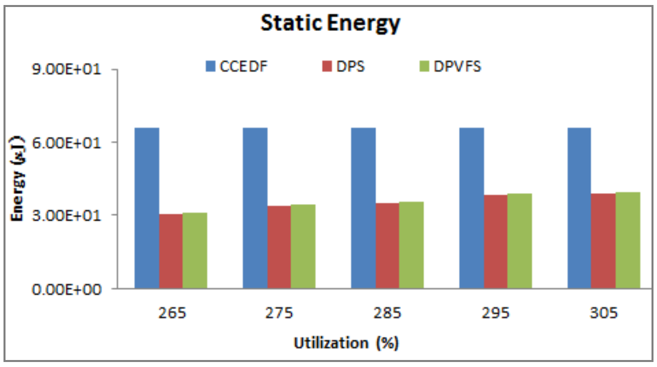


Figure 4 Dynamic energy consumption per unit time for different utilisations (see online version for colours)

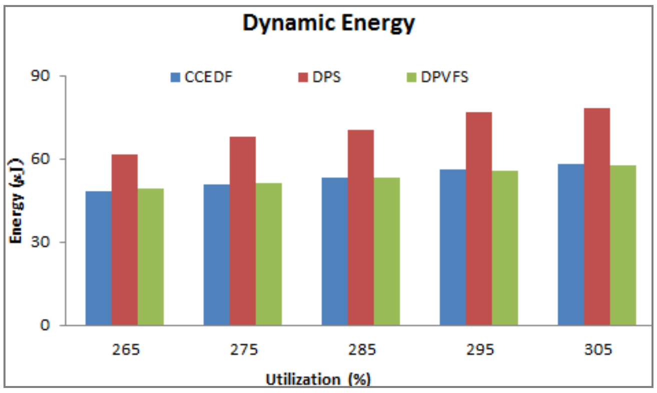


Figure 4 shows the dynamic energy consumption per unit time for different utilisations. Irrespective of the utilisations, CCEDF offers the least dynamic energy consumption followed by DPVFS and DPS algorithms. The DPVFS algorithm offers very close performance in comparison with CCEDF which is because of its aggressive shutdown and slowdown strategies. DPVFS increases dynamic energy by 0.38% over CCEDF and saves 32.7% over DPS.

Figure 5 Decision making energy per unit time for different utilisations (see online version for colours)

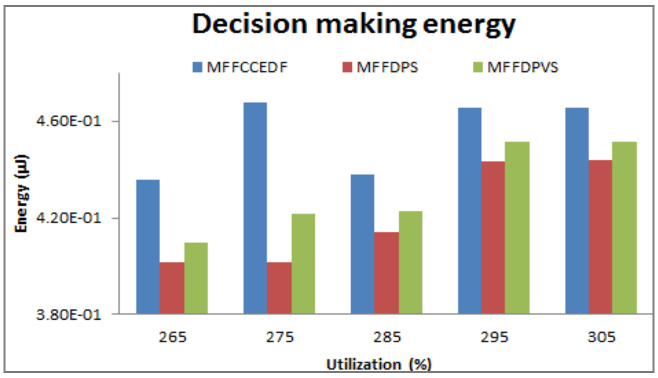
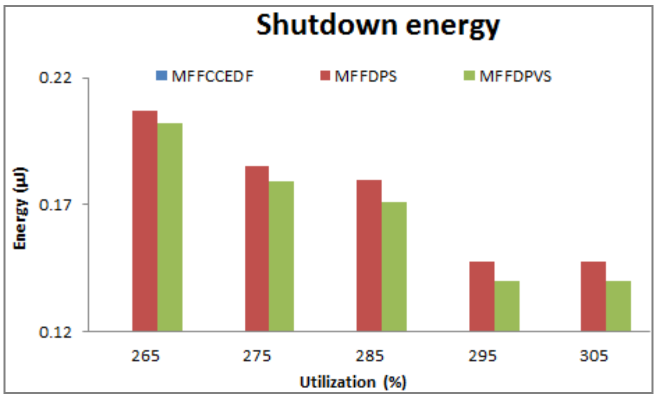


Figure 5 shows the decision making energy per unit time for different utilisations. In procrastination with voltage scaling approach, at the beginning of every active period, current active duration and next available idle duration is computed. Based on this idle duration, active period and scaled voltage is computed. The idle duration computation is also carried out when there are no jobs in ready queue. This adds into the decision making overhead. Thus DPVFS consumes more energy in decision making compared to DPS. In CCEDF, at every arrival and completion of job, the scaled voltage is computed and thus has more number of decision points compared to DPVFS. DPVFS has decision making overhead of 2.44% over DPS and 5.38% saving over CCEDF respectively.

Figure 6 shows the decision making energy per unit time for different utilisations. The chance of converting inactive period into active period by voltage scaling approach lessens the chance of shutting down the core. DPVFS reduces the shutdown overhead by 4.43% over DPS.

Figure 6 Shutdown energy per unit time for different utilisations (see online version for colours)



Though the decision making and context switching energy are less in procrastination schedulers, it is not significant as compared to static and dynamic energy reductions.

Figure 7 Idle energy consumption per unit time for different utilisations (see online version for colours)

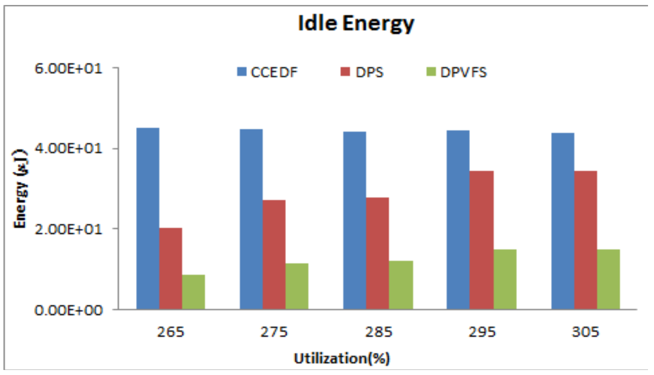


Figure 7 shows the idle energy consumption per unit time for different utilisations. DPVFS consumes the least idle energy followed by DPS and CCEDF. This is because DPVFS converts most of the inactive periods into either active or shutdown and leaves much less idle duration whereas CCEDF converts inactive period into active or idle period and DPS converts inactive period into shutdown wherever possible. On an average CCEDF and DPS produces 72% and 56.8% more idle energy respectively than DPVFS.

Figure 8 Total energy consumption per unit time for different utilisations (see online version for colours)

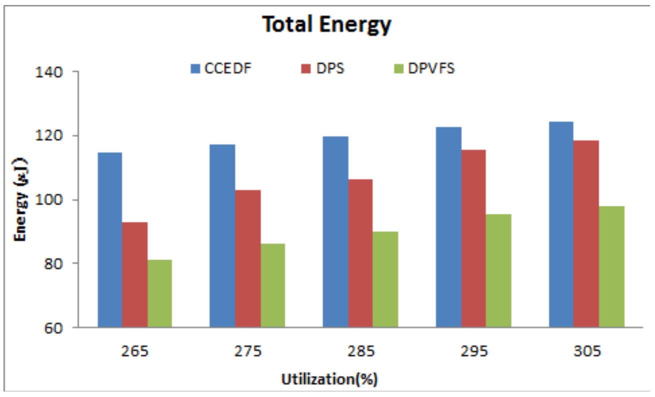


Figure 8 shows the total energy consumption per unit time for different utilisations with shutdown threshold as 500 time units. DPVFS consumes least energy compared to CCEDF and DPS. With the increase in utilisation for same shutdown threshold, the chance of shutting down the core reduces since active duration increases. This increases the static energy consumption in DPS algorithm and both static and dynamic energy consumption in CCEDF compared to DPVFS algorithm. Thus DPVFS gives significant saving in total energy compared to DPS algorithm. On an average DPVFS reduces total energy by 33.2% and 18.8% over CCEDF and DPS respectively.

Figure 9 Total energy consumption per unit time for different shutdown thresholds (see online version for colours)

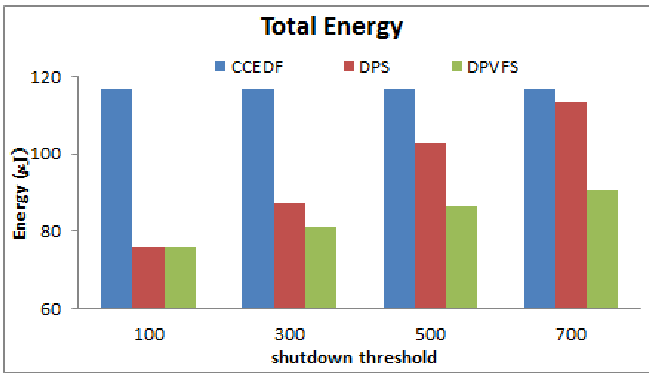


Figure 9 shows the total energy consumption per unit time with different shutdown thresholds for task set having 275% utilisation. With the increase in threshold, DPVFS offers significant saving in total energy compared to DPS algorithms. CCEDF has no impact with shutdown threshold as it never allows the processor to shutdown. DPVFS saves 54.14% and 0.15% of energy over CCEDF and DPS when shutdown threshold is set to 100. It saves 28.84% and 25.06% of energy over CCEDF and DPS when shutdown threshold is set to 700.

Figure 10 Total, static and dynamic energy consumption per unit time for different number of cores (see online version for colours)

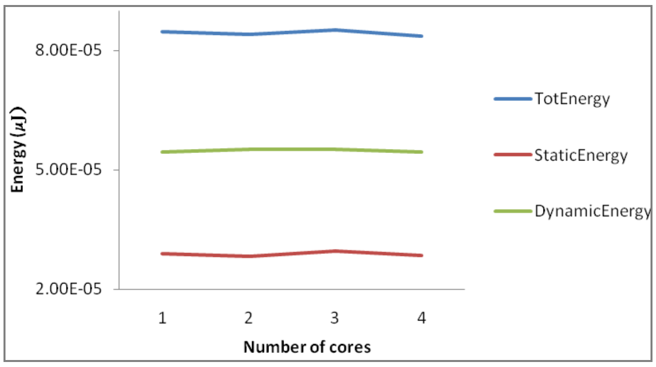


Figure 10 shows the total, static and dynamic energy consumption per unit time for different number of cores. In spite of increase in number of cores, the total, static and dynamic energy consumption of DPVFS remains constant when each core maintains on an average same utilisation.

9 Conclusions and future scopes

This paper proposed DPVFS to address an efficient way of reducing energy consumption while scheduling the real-time tasks in multi-core system. The DPVFS is a dynamic real-time priority scheduling algorithm which reduces static and dynamic energy consumption of the executing task set. It first does an efficient task allocation to improve the possibility of shutdown. It then schedules the tasks efficiently to achieve maximum shutdown and minimum idle duration. This work compares the static, dynamic, idle and total energy consumption using DP, DVFS and DPVFS approaches. Experimental results show that the DPVFS saves 33.2% and 18.8% of total energy over CCEDF and DPS respectively. DPVFS achieved 84.54% of static energy saving and up to 0.38% increase in dynamic energy over CCEDF. Compared to DPS, the static energy in DPVFS increases by 1.45% but it reduces the dynamic energy by 32.7%. DPVFS saves the idle energy consumption by 271.5% and 131.4% over CCEDF and DPS respectively. DPVFS causes decision making overhead of 2.44% over DPS and 5.38% saving over CCEDF respectively. DPVFS reduces shutdown overhead by 4.43% over DPS. It is observed that FF task allocation with period as the sorting base, DPVFS task scheduling in MC systems offers better energy saving over all other algorithms. Schedulability of tightly coupled multi-core systems can be improved with the help of migrations. Migration can also be used effectively for reducing energy consumption. We will be experimenting the impact of migration both for DVFS and for DP as future work. We also would investigate the impact of sharing resource usage in multi-core DVFS and DP.

References

Awan, M. and Petters, S. (2011) ‘Enhanced race-to-halt: a leakage-aware energy management approach for dynamic priority systems’, in *2011 23rd Euromicro Conference on Real-Time Systems*, IEEE, pp.92–101.

Aydin, H., Melhem, R., Mossé, D. and Mejía-Alvarez, P. (2001) ‘Dynamic and aggressive scheduling techniques for power-aware real-time systems’, in *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, Cat. No. 01PR1420, IEEE, pp.95–105.

Aydin, H., Melhem, R., Mossé, D. and Mejía-Alvarez, P. (2004) ‘Power-aware scheduling for periodic real-time tasks’, in *IEEE Transactions on Computers*, IEEE, pp.584–600.

Chen, G., Huang, K. and Knoll, A. (2017) ‘Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination’, *ACM Transactions on Embedded Computing Systems (TECS) – Special Issue on Design Challenges for Many-Core Processors*, Special Section on ESTIMedia’13 and Regular Papers, Vol. 13, pp.1–2, ACM.

Chen, J. and Kuo, T. (2006) ‘Procrastination for leakage-aware rate monotonic scheduling on a dynamic voltage scaling processor’, in *ACM SIGPLAN 2006 on Language, Compilers and Tool Support form Embedded Systems*, ACM, pp.153–162.

Devdas, V. and Aydin, H. (2012) 'On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications', in *IEEE Transactions on Computers*, Vol. 61, pp.31–44, IEEE.

Gabriel, P. and De Mello, R. (2015) 'Modelling distributed computing workloads to support the study of scheduling decisions', *Int. J. Comput. Sci. Eng.*, September, Vol. 11, No. 2, pp.155–166, ISSN: 1742-7185, DOI: 10.1504/IJCSE.2015.071879 [online] http://dx.doi.org/10.1504/IJCSE.2015.071879 (accessed 30 June 2017).

Gawali, S. and Raveendran, B. (2016) 'DPS: a dynamic procrastination scheduler for multi-core/multi-processor hard real-time systems', in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, pp.286–291.

Gu, J. and Qu, G. (2013) 'Incorporating temperature-leakage interdependency into dynamic voltage scaling for real-time systems', in *2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors*, June, pp.289–296, DOI: 10.1109/ASAP.2013.6567592.

Hu, W., Chen, T., Xie, B. and Yan, L. (2010) 'Embedded hard real-time scheduling algorithm based on task's resource requirement', *Int. J. High Performance Computing and Networking*, Vol. 6, Nos. 3/4, pp.234–239 [online] http://dx.doi.org/10.1504/IJHPCN.2010.037798 (accessed 30 June 2017).

Huang, K., Santinelli, L., Chen, J., Thiele, L. and Buttazzo, G. (2009) 'Periodic power management schemes for real-time event streams', in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, IEEE, pp.6224–6231.

Jejurikar, R. and Gupta, R. (2005) 'Dynamic slack reclamation with procrastination scheduling in real-time embedded systems', in *42nd IEEE Design Automation Conference*, IEEE, pp.111–116.

Jejurikar, R., Pereira, C. and Gupta, R. (2004) 'Leakage aware dynamic voltage scaling for real-time embedded systems', in *41st IEEE Design Automation Conference*, IEEE, pp.275–280.

Khaleel, M. and Zhu, M. (2016) 'Energy-efficient task scheduling and consolidation algorithm for work flow jobs in cloud', *Int. J. Comput. Sci. Eng.*, January, Vol. 13, No. 3, pp.268–284, ISSN: 1742-7185, DOI: 10.1504/IJCSE.2016.078933 [online] http://dx.doi.org/10.1504/IJCSE.2016.078933 (accessed 30 June 2017).

Lee, Y., Reddy, K. and Krishna, C. (2003) 'Scheduling techniques for reducing leakage power in hard real-time systems', in *15th IEEE Euromicro Conference on Real-Time Systems*, IEEE, pp.105–112.

Legout, V., Jan, M. and Paulet, L. (2013) 'A scheduling algorithm to reduce the static energy consumption of multiprocessor real-time systems', in *21st ACM International conference on Real-Time Networks and Systems, RTNS 2013*, ACM, pp.99–108.

Lin, H-Y., Doong, J-G., Hsieh, M-Y. and Li, K-C. (2017) 'A real-time vehicle management system implementation on cloud computing platform', *Int. J. of High Performance Computing and Networking*, Vol. 10, No. 3, pp.168–178.

Liu, J. (2004) *Real-Time Systems*, 5th ed., Pearson Education, India, ISBN: 81-7808-463-5.

Liu, Y., Lin, M., Taniguchi, I. and Tomiyama, H. (2017) 'A dual-mode scheduling approach for task graphs with data parallelism', *Int. J. of Embedded Systems*, Vol. 9, No. 2, pp.147–156.

Meisner, D., Gold, B. and Wenisch, T. (2009) 'Powernap: eliminating server idle power', in *SIGARCH Comput. Archit. News*, March, Vol. 37.

Nassiffe, R., Camponogara, E., Lima, G. and Mossé, D. (2016) 'Optimising QoS in adaptive real-time systems with energy constraint varying CPU frequency', Inderscience Publishers, Geneva, Switzerland, pp.368–379 [online] http://dx.doi.org/10.1504/IJES.2016.10001303 (accessed 30 June 2017).

Niu, L. and Quan, G. (2009) 'Reducing both dynamic and leakage energy consumption for hard real-time systems', in *ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems, CASES 2004*, ACM, pp.140–148.

Pagani, S. and Chen, J.J. (2013) 'Energy efficiency analysis for the single frequency approximation (SFA) scheme', in *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, IEEE, pp.82–91.

Pillai, P. and Shin, K. (2001) 'Real-time dynamic voltage scaling for low-power embedded operating systems', in *ACM SIGOPS Operating Systems Review*, ACM, Vol. 35, pp.89–102.

Prakash, S., Trivedi, V. and Ramteke, M. (2016) 'An elitist non-dominated sorting bat algorithm NSBAT-II for multi-objective optimization of phthalic andride reactor', Inderscience Publishers, Geneva, Switzerland, Vol. 7, pp.200–315.

Renaux, D. (2014) 'Comparative performance evaluation of CMSIS-RTOS', in *2014 Brazilian Symposium on Computing Systems Engineering*, Inderscience Publishers, November, pp.126–131, DOI: 10.1109/SBESC.2014.9.

Shajulin, B., Rejitha, R.S., Preethi, C., Bency Bright, C. and Judyfer, W.S. (2017) 'Energy analysis of code regions of HPC applications using energyanalyzer tool', *Int. J. of Computational Science and Engineering*, Vol. 14, No. 3, pp.267–278.

Shiv, P. and Deo, P. (2014) *A Hybrid Immune Genetic Algorithm for Scheduling in Computational Grid*, Inderscience Publishers, Geneva, Switzerland, Vol. 6.

Trivedi, V., Prakash, S. and Ramteke, M. (2016) 'Optimized online control of MMA polymerization using fast multi-objective DE', *Taylor and Francis Online*, pp.1–8.

Wu, J. (2014) 'Energy-efficient scheduling of real-time tasks with lock-free objects', in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, Inderscience Publishers, Geneva, Switzerland, August, pp.225–230, DOI: 10.1109/DASC.2014.48.

Yang, C., Chen, J. and Kuo, T. (2005) 'An approximation algorithm for energy-efficient scheduling on a chip multiprocessor', in *IEEE Design, Automation and Test in Europe*, IEEE, pp.468–473.