ISSN 2347 - 3983
Volume 8. No. 9, September 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter157892020.pdf
https://doi.org/10.30534/ijeter/2020/157892020

A Multi-Speed Dynamic Voltage/Frequency Scaling Scheme for Multicore Real-Time Systems

Sangchul Han, Jinsoo Kim, Junghwan Kim*
Dept. of Software Technology,Konkuk University, Korea
*Corresponding author: jhkim@kku.ac.kr

ABSTRACT

DVFS (dynamic voltage /frequency scaling) is a well-known and prevalent low power technique in processor components. This article proposes a DVFS scheme for multicore real-time systems. The proposed scheme divides the worst-case processor time of a real-time task into multiple sections and associates different speed with each section in increasing order. As a job executes and progresses, its speed increases according to the speed associated with the sections. If a job's actual execution requirement is smaller than its worst-case execution requirement, the job starts and executes at a low speed. The amount of consumed energy is less than the worst case. We integrate and evaluate the scheme with a multicore real-time scheduling. The results show that our multi-speed scheme can save energy consumption by up to 6.99% compared with the *individual speed scheme*. And we also show that our scheme performs better as the number of sections increases.

Key words : Multi-speed DVFS, Multicore, EDZL Scheduling, Real-Time Systems

1. INTRODUCTION

The limitation of battery capacity forces embedded system designers to devise low power techniques in hardware or software components[23]-[27].DVFS (dynamic voltage/frequency scaling) is a well-known and prevalent low power technique in processor components. By decreasing the frequency and supply voltage dynamically in processors, DVFS can reduce the power dissipation in processors. Many processor manufacturers incorporate DVFS technique into their products[3]-[5]. The challenge of adopting DVFS in embedded real-time systems is that decreasing the frequency and supply voltage causes processing cores to execute slow. It takes a longer time for a real-time task to fulfil its computation than no DVFS technique is adopted. If the core speed is excessively lowered, real-time tasks may not finish in time, resulting in deadline miss.

There have been many studies on DVFS on multiprocessor (or multicore) real-time systems [1,6-16,22,26-27]. Among them, Han et. al [22] proposes the *individual speed scheme* that calculates an individual speed for periodic tasks in EDZL [17-21] scheduling on multicore platforms. The authors showed that tasks running at their individual speed can be safely scheduled to meet their deadline and consume much less energy.

In this article, we propose a novel DVFS scheme where tasks have multiple steps of speed. From the individual speed for EDZL scheduling, we can calculate the processor time that can be safely allowed to a task without jeopardizing other tasks' timing constraints. We can adjust the task's execution speed to reduce energy consumption as long as it fulfils its execution within that processor time. Our scheme divides the processor time into multiple sections and associates different speed to each section in increasing order. A job starts execution at a speed lower than its individual speed. As the job progresses, its speed increases according to the speed associated with the sections. If the AET (actual execution time) of the job is far smaller than its WCET (worst-case execution time), the energy consumption is less than executing all the computation at the individual speed. We integrate and evaluate the proposed scheme with EDZL scheduling. The results show that our scheme can save energy consumption by up to 6.99% compared with the individual speed scheme. And we also show that our scheme performs better as the number of sections increases.

2. SYSTEM MODEL

Multicore platforms consist of *m* cores that have an individual clock. The core speed is defined as$f$ / f_{max} where f is the current frequency and f_{max} is the maximum frequency. T he minimum frequency is denoted by f_{min} and the minimum core speed is denoted by S_{min}. Obviously, the maximum core speed is 1. The execution amount of job is defined as $l \times s$ where l is the used processor time and s is the core speed. If a job executes from t_1 to $t_2$atspeed0.7, the amount of execution is $(t_2 - t_1) \times 0.7$.

Assume $P=V^2 f$ where P is the power dissipation of core and V is the supply voltage [2]. As usual, to adjust the speed of core, the voltage and frequency of core should be changed together. In addition, since V and s are nearly proportional to f, we can say that P is proportional to the cubic of s. Since the length of used processor time of job is inversely proportional to its speed for a fixed execution amount, the energy consumption of job is proportional to s^2.

A periodic task τ_i is (e_i, p_i) where p_i is a period and e_i is the WCET supposing τ_i executes at the maximum speed. A task periodically instantiates job $\tau_{i,j}$ at $j \cdot p_i$ (j=0,1,2,…) and $\tau_{i,j}$ requires the execution ofat most e_i time unit. The job has to fulfil the execution by deadline $d_{i,j} = p_i(j+1)$.A task set is denoted by $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. The total utilization of taskset is $U(\tau) = \Sigma(e_i/p_i)$.

3. MULTI-SPEED DVFS SCHME

3.1 Individual Speed Technique

Han et. al [22] proposed a DVFS technique called the individual speed scheme. Based on Lee's test [21], it computes an individual speed for each task. From the individual speed, we can calculate the processor time that can be safely allowed to a task without jeopardizing other tasks' timing constraints. We can adjust the execution speed of the task to reduce energy consumption as far as it fulfils its execution within that processor time. Let S_i be the individual speed of τ_i. The amount of processor time that τ_i can use is e_i/S_i in every period.

3.2Multi-Speed DVFS Technique

This section proposes the multi-speed DVFS technique. Our scheme divides the processor time allowed to τ_i into multiple sections of the same length. Let N be the number of sections. The length of each section is $e_i/(S_iN)$. The technique associates different speed with each section in increasing order.

For tasks with individual speed higher than or equal to 0.5, the execution speed from $2\times S_i$-1 to 1.0 with a step of $(2-2\times S_i)/(N-1)$is assigned to the sections. The speed assigned to the first section is lower than the individual speed S_i by 1-S_i.Since the length of sections are all equal and the speed of sections are symmetric with respect to S_i, the total execution amount of task in our scheme is equal to that in the individual speed scheme. Hence, jobs can fulfil its execution in the worst case.

Jobs start execution at a speed lower than its individual speed. As jobs progress, its speed is raised according to the speed of the sections. If the AET of a job is far smaller than its WCET, since the job starts execution at a low speed, the energy consumption is less than executing all the computation at its individual speed. Figure 1 illustrates the multi-speed scheme when the number of sections is 4, i.e.,N=4. The individual speed of this task is 0.7. The speeds of the sections are 0.4, 0.6, 0.8 and 1.0. Jobs of this task start execution at speed 0.4. The speed is raised to 0.6 when jobs has executed for totally $e_i/(4S_i)$ time unit on cores. Suppose a job finishes in the second section. Since the speed of the first and second section is lower than the individual speed, the energy consumption is less than it would be with the individual speed.

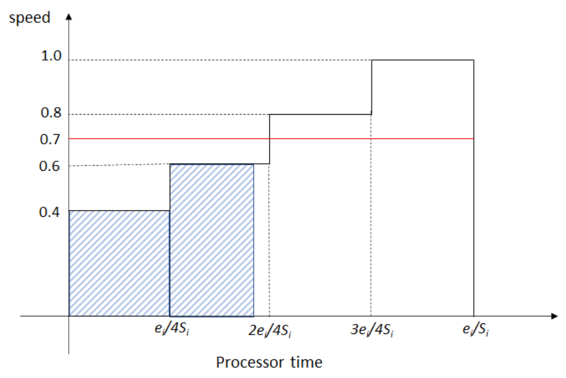

Figure 1: Example for multi-speed scheme when N=4

For tasks with individual speed lower than 0.5, the scheme does not apply multi-speed DVFS technique. Those tasks execute at their individual speed until they finish. In our scheme, the individual speed is the middle of the section speeds. If the individual speed is close to the minimum speed, there is little room in the space of speed for the head sections. For example, the minimum speed of Strong ARM SA-1100 processor is 0.291. If the individual speed is 0.4, the difference between them is 0,109. The speed of jobs can be lowered just a little.

4. EXPERIMENTS

We conduct simulation to evaluate the multi-speed DVFS scheme. Groups of tasksets are created for each m = 2, 4,8 and16. As we vary the total utilization from 0.25m to 0.90m with a step of 0.2, we create 100 tasksets for each utilization. Each taskset is composed of a random number of tasks. When generating a task, we randomly select p_i from (10,1000] and u_i from (0.1,1],and calculate $e_i=p_i \times u_i$. We test the tasksets using Lee's test [21] and drop unschedulable tasksets. The actual execution time of jobs is randomly selected from [1, e_i].

The processor characteristics in our experiments are shown in Table 1. Every time a job starts a section, the amount of processor time that is used in this section is calculated and the amount of energy consumption is also calculated using the processor characteristics corresponding to the section speed. The energy consumption is $E=P\times l$ where l is the length of execution and P is the power in Table 1. While a job executes, the energy consumed during its execution is summed up.

Table 1: Characteristics of StrongARM SA-1100

power(%)	speed	volt.(V)	freq.(MHz)
100	1.000	1.50	206
78.9	0.947	1.42	195
63.2	0.874	1.30	180
50.0	0.801	1.20	165
39.9	0.728	1.15	150
33.6	0.655	1.10	135
33.3	0.583	1.08	120
19.8	0.510	0.95	105
15.0	0.437	0.90	90
11.8	0.364	0.82	75
9.44	0.291	0.80	60

The energy consumption of a taskset is the sum of the energy consumed by all jobs belonging to the taskset for a hyper-period. We compute the ratio of the energy consumption with a DVFS scheme to the energy consumption with no DVFS scheme for each taskset. Then we average the ratio over total utilizations.

The average energy consumption ratio is shown in Figure2, 3, 4, and5. In the figures, *individual* denotes the individual speed scheme [22] and *N-step* denotes the proposed scheme with the number of sections of *N*(=2, 3, 4, 5 and 6).In case *N*=2, our scheme consumes less energy than *individual* only for large total utilizations. If the total utilization is not high, many tasks have a small execution requirement. For such tasks, the difference between the AET and the WCET is small, resulting in small energy saving gain from a low execution speed. When such a job enters the second section, the execution speed jumps up to the maximum speed, resulting in much energy consumption. We can find out from the figures that the energy consumption reduces as the number of sections increases. This means that moderate speed increase is more effective in energy saving. Our scheme performs better as the number of sections increases. It can save energy consumption by up to 6.99% compared with the individual speed scheme when *m*=2 and *N*=6.

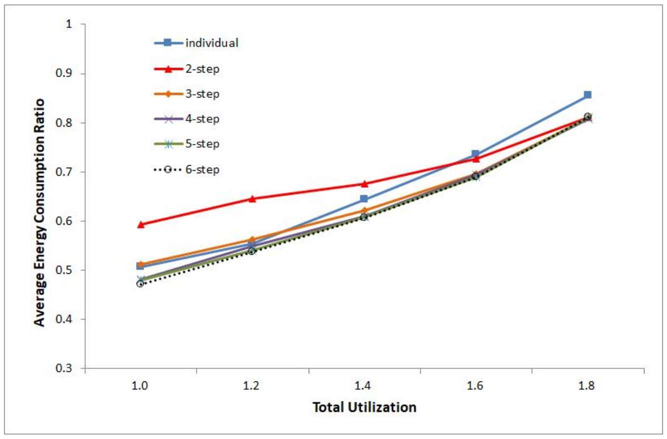

Figure 2: Average energy consumption ratio on 2 cores

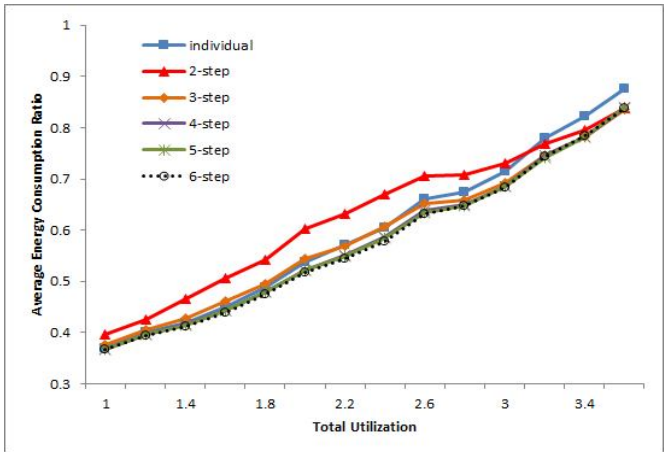

Figure 3: Average energy consumption ratio on 4 cores

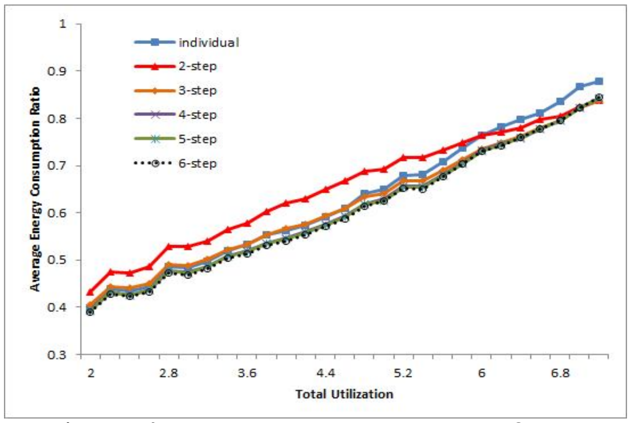

Figure 4: Average energy consumption ratio on 8 cores

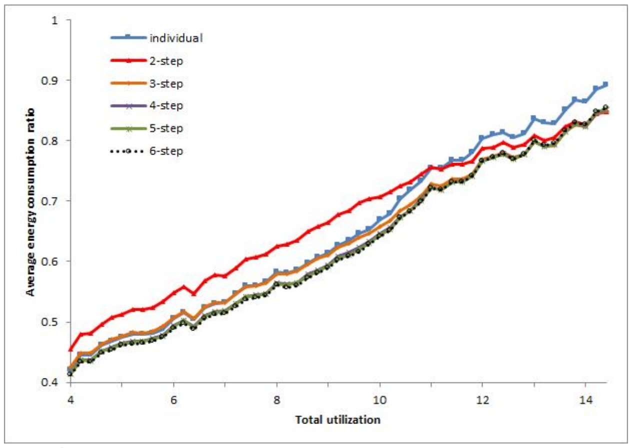

Figure 5: Average energy consumption ratio on 16 cores

Figure 6, 7, 8 and 9 demonstrate the relative performance of our multi-speed scheme compared with the individual speed scheme. The figures show the average energy consumption ratio normalized to the individual speed scheme. With a small number of sections, our scheme underperforms the individual speed scheme significantly. For example, the difference is 6.49%. when *m*=2 and *N*=2. However, as the number of sections increases, the performance of our scheme improves. When *m*=2 and *N*=6, our scheme outperforms the individual speed scheme by 5.44%.

5. CONCLUSION

This article proposes a multi-speed dynamic voltage/frequency scaling scheme for EDZL scheduling algorithm. Our scheme divides the allowed processor time of tasks into several sections and assigns an execution speed to the sections in increasing order. Since the assigned speeds are symmetric with respect to the individual speed, we can safely adjust a job's speed without jeopardizing the timing constraints. By starting a job at a speed lower than its individual speed, we can expect energy saving if the actual execution requirement is much less than the worst case. The simulation results show that our scheme underperforms the individual speed scheme when the number of sections is small. However, as the number of sections increases, that is, the execution speed is adjusted more moderately, the performance of our scheme improves. It can save more energy than the individual speed scheme by at most about 7%.

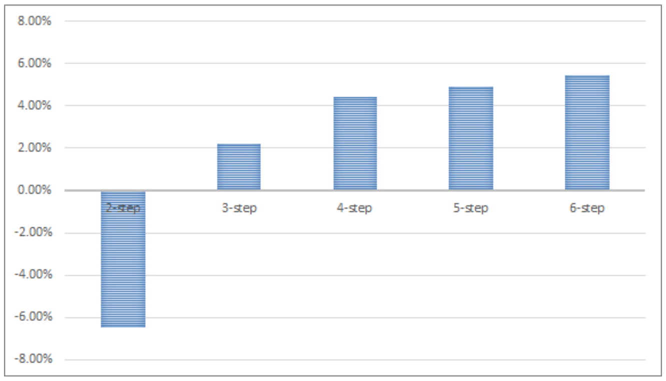

Figure 6: Relative performance of our scheme on 2 cores

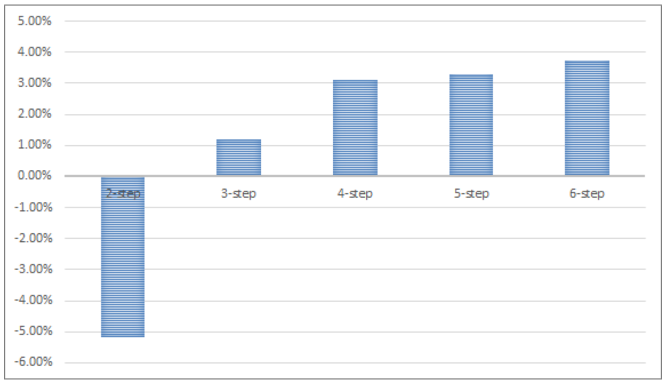

Figure 7: Relative performance of our scheme on 4 cores

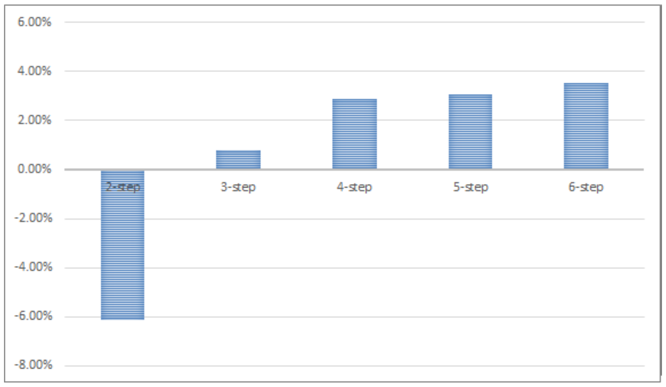

Figure 8: Relative performance of our scheme on 8 cores

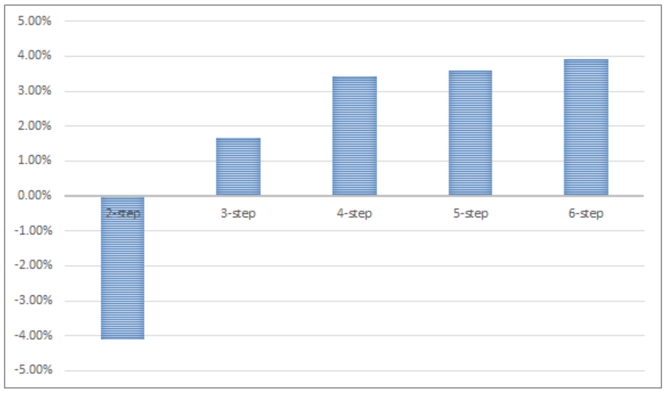

Figure 9. Relative performance of our scheme on 16 cores

REFERENCES

1. V. Nelis, J. Goossens, R. Devillers, and N. Navet, **Power-aware real-time scheduling upon identical multiprocessor platforms**, in *Proc. IEEE International conference on sensor networks, ubiquitous and trustworthy computing (SUTC'08)*, pp. 209–216, 2008.
2. A. Chandrakasan, S. Sheng, and R. Brodersen. **Low-Power CMOS Digital Design**, *IEEE Journal of Solid-State Circuit*, vol. 27, no. 4, pp. 473–484, 1992.
3. Intel Product Specifications. Retrieved November 29, 2018, from: https://ark.intel.com/.
4. J. Khan, S. Bilavarn, and C. Belleudy. **Energy Analysis of a DVFS based power strategy on ARM platforms**, in *Proc. IEEE Faible Tension Faible Consommation (FTFC)*, Paris, France, pp. 1–4, 2012.
5. AMD Products. Retrieved November 29, 2018, from: http://www.amd.com/en-us/products.
6. H. Aydin, and Q. Yang, Q. **Energy-aware partitioning for multiprocessor real-time systems**,in *Proc. 17th International Symposium on Parallel and Distributed Processing (IPDPS'03)*, Nice, France, 2013.
7. J.J.Chen, and T.W. Kuo. **Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics**, in *Proc. the 2005 International Conference on Parallel Processing (ICPP'05)*, Oslo, Norway, pp. 13–20, 2005.
8. C.Y. Yang, J.J. Chen, and T.W Kuo. **An approximation algorithm for energy-efficient scheduling on a chip multiprocessor**. in *Proc. Conference on Design, Automation and Test in Europe (DATE'05)*, Munich, Germany, pp. 468–473, 2005
9. J.J. Chen, and C.F. Kuo. **Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms**, in *Proc. 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07)*, Daegu, Korea, pp. 28–38, 2007.
10. C.L. Liu, and J.W. Layland. **Scheduling Algorithms for Multiprogramming in a HardReal-Time Environment**, *Journal of the ACM*. vol. 20, no. 1, pp. 46–61, 1973.
11. N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. **Hard real-Time scheduling: the deadline-monotonic approach**, in *Proc. IFAC/IFIP Workshop on Real Time Programming*, Atlanta, USA, pp. 127-132, 1991.
12. K. Funaoka, A. Takeda, S. Kato, and N. Yamasaki. **Dynamic Voltage and Frequency Scaling for Optimal Real-Time Scheduling on Multiprocessors**, in *Proc. 3rd IEEE International Symposium on Industrial*

Embedded Systems (SIES'08), Le Grande Motte, France, pp. 27–33, 2008.

13. V. Nelis, J. Goossens, R. Devillers, and N. Navet. **Power-Aware Real-Time Scheduling upon Identical Multiprocessor Platforms**, in *Proc. IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing (SUTC'08)*, Taichung, Taiwan, pp. 209–216, 2008.
14. X. Piao, H. Kim, Y. Cho, S. Han, M. Park, and M. Park. **Power-Aware EDZL Scheduling upon Identical Multiprocessor Platforms**, in. *Proc. International Conference on Reliable and Autonomous Computational Science (RACS 2010)*, Atlanta, USA, pp. 61–80, 2010.
15. S. Funk, V. Berten, C. Ho, and J. Goossens. **A global optimal scheduling algorithm for multiprocessor low-power platforms**, in *Proc. 20th International Conference on Real-Time and Network Systems*, Pont à Mousson, France,pp. 71–80, 2012.
16. S. Han, M. Park, X. Piao, and M. Park. **A dual speed scheme for dynamic voltage scaling on real-time multiprocessor systems**, *The Journal of Supercomputing*,vol. 71, no. 2, pp. 574–590, 2015.
17. S. Cho, S.K. Lee, A. Han, and K.J. Lin. **Efficient Real-Time Scheduling Algorithms for Multiprocessor Systems**,*IEICE Trans on Communications*, vol. E85-B, no. 12, pp. 2859–2867, 2002.
18. M. Park, S. Han, H. Kim, S. Cho, and Y. Cho. **Comparison of Deadline-based Scheduling Algorithms for Periodic Real-Time Tasks on Multiprocessor**,*IEICE Trans on Information and Systems*,vol. E88-D, no. 3, pp. 658–661, 2005.
19. M. Cirinei, and T.P. Baker. **EDZL Scheduling Analysis**,in *Proc. 19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, Pisa, Italy, pp. 9–18, 2007.
20. T.P. Baker, M. Cirinei, and M. Bertogna. **EDZL scheduling analysis**, *Real-Time Systems,*vol. 40, no. 3, pp. 264–289, 2008.
21. J. Lee, and I. Shin. **EDZL Schedulability Analysis in Real-Time Multicore Scheduling**, *IEEE Transactions on Software Engineering*, vol. 39, no. 7, pp. 910–916, 2013.
22. S. Han, M. Park, and W. Paik.**Dynamic Voltage/Frequency Scaling for EDZL Scheduling in Multicore Real-Time Systems**, *Journal of Engineering and Applied Sciences*, vol. 14, no. 21, pp. 8039-8046, 2019.
23. M S. Kumar, F. Noorbasha, S. Inthiyaz, M. Jameela, A. Sandhya, Md. Imran, and S. K. Tulasi. **Low Power Carry Look-Ahead Adder using Transmission Gate Multiplexer**, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 1, 2020.
24. B. Lakshmi, and B. Navyasri. **Energy Efficient Routing Mechanism for Harsh Environment in Wireless Sensor Networks**, *International Journal of Emerging Trends in Engineering Research*, vol. 7, no. 9, 2019.
25. P.S. Akram, G.V. Ganesh, A. S. Kumar, K.S. Chand, and M.R. Varma. **Non-Volatile 7T1R SRAM cell design for low voltage applications**, *International Journal of Emerging Trends in Engineering Research*, vol. 7, no. 11, 2019.
26. S. Han. **Energy-aware EDZL Scheduling of Periodic Tasks on Multicore Systems**, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 4, 2020.
27. S. Han. **A Simple and Aggressive Dynamic Voltage/Frequency Scaling Technique for EDZL Scheduling on Multiprocessors**, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 8, 2020.