ISSN 2347 - 3983
Volume 8. No. 4, April 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter18842020.pdf
https://doi.org/10.30534/ijeter/2020/18842020

Energy-aware EDZL Scheduling of Periodic Tasks on Multicore Systems

Sangchul Han
Dept. of Software Technology, Konkuk University, Korea, schan@kku.ac.kr

ABSTRACT

DVFS (Dynamic Voltage/Frequency Scaling) has been a significant technique for reducing energy consumption of processors. Energy-aware scheduling schemes make use of DVFS feature of processors to adjust the execution speed of processors depending upon workload; high speed for heavy workload and low speed for light workload. This paper proposes a combined DVFS scheme for EDZL (Earliest Deadline until Zero Laxity)scheduling. The scheme combines a per-core DVFS technique that calculates tasks' individual speed and MOTE technique that lowers the execution speed of a core by reclaiming slack time. Experiment results show that our scheme can decrease the energy consumption in executing periodic task.

Key words : DVFS, EDZL Scheduling, Embedded Systems, Multicore, Periodic Task

1. INTRODUCTION

Reducing energy consumption has been a significant concern in recent computing systems including mobile embedded systems as well as cloud systems. There are many energy saving techniques for various computer components [23]-[25] such as PAVM (Power-Aware Virtual Memory),DPM (dynamic power management), and DVFS (dynamic voltage/frequency scaling). Among them DVFS is an energy-saving technique for processors that adjusts processor supply voltage and/or frequency according to the workload; lowering the supply voltage/frequency on light workload reduces the energy consumption of processors. The DVFS technology developed by processor benders are EIST(Enhanced Intel SpeedStep) by Intel [3], IET (Intelligent Energy Management) by ARM [4], and PowerNow! by AMD [5].

The problem of deploying DVFS in scheduling real-time tasks is that a lower supply voltage/frequency may cause real-time tasks to violate their timing constraints, i.e., to miss their deadline. Thus, energy-aware scheduling schemes should make use of DVFS feature of processors carefully.

Many researchers have studied DVFS techniques for real-time scheduling on multicore systems. Some developed DVFS techniques in partitioning approach[6]-[9], where tasks are partitioned into groups and each task group is executed on a processing core statically. In partitioning approach, a uniprocessor scheduling algorithm [10][11] is employed on each core. Some researchers devised DVFS techniques in global approach [12]-[16], where tasks can start or resume on any available processing core.

Earliest Deadline until Zero Laxity (EDZL) [17] is a global real-time scheduling algorithm. It assigns the highest priority to jobs with zero laxity. It gives to the remaining jobs priority according to EDF (Earliest Deadline Frist). EDZL dominates global EDF [18], and is superior to other EDF variants [19]-[21].There are some research works on DVFS techniques for EDZL algorithm. Piao et al. [14] proposed a DVFS technique for EDZL that calculates a static uniform speed; at any time all cores execute periodic tasks at the speed. Han et al. [22] proposed energy-aware EDZL scheduling that computes static individual speed for each periodic task on per-core DVFS platforms; all cores can execute periodic tasks at different speed.

In this paper, we propose an aggressive DVFS technique for EDZL scheduling algorithm on multicore platforms. By combining Han's DVS technique with MOTE [1] which is an on-line DVS technique, our scheme can safely reduce the execution speed of cores. The experiment results show that our scheme can further reduce the energy consumption of periodic tasks compared with Han's technique. It can reduce more energy by %.

2. SYSTEM MODEL

2.1 Processor Model

We assume that there are$m$ identical processing cores P_1, P_2, …, P_m and each core contains an individual clock, i.e., per-core DVFS platforms where the clock frequency of each core can be adjusted individually. The speed of core $P_i$is denoted by $s_i = f / f_{max}$ where f is the current frequency and f_{max} is the maximum frequency of P_i. The range of s_i is $[s_{min}, s_{max}]$ where $s_{min}$is the minimum speed and s_{max}(=1) is the maximum speed. For example, if a core executes a job from time t_1 to time t_2 with a clock frequency $0.5 \times f_{max}$, the speed of the core during $[t_1, t_2)$ is 0.5 and the amount of execution is $(t_2 - t_1) \times 0.5$.

We assume that a core's power dissipation is$P \propto V^2 f$,where V is the supply voltage and f is the frequency of the core [2]. Since it is assumed that $f \propto V$and the core speed s is

proportional to f, we have $P \propto s^3$[2]. For example, if a core executes a job during $[t_1, t_2)$ at speed s, the amount of energy consumption of the job is $(t_2 - t_1) \times s^3$.

2.2 Task Model

A tasks system consists of n periodic tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task is denoted by $\tau_i = (e_i, p_i)$ where p_i is a period and e_i is the worst-case execution time assuming the task executes at s_{max}; τ_i generates a job $\tau_{i,j}$at time $j \cdot p_i$ (j=0,1,2,…) which requires the execution of e_i time units at most, and $\tau_{i,j}$ should be finished its execution by absolute deadline $d_{i,j} = p_i(j+1)$. The utilization of τ_i denoted by u_i is e_i/p_i and the total utilization of τdenoted by $U(\tau)$ is $\Sigma_{\tau i \in \tau} u_i$. We define $U_{max}(\tau) = max\{u_i | \tau_i \in \tau\}$.

Let$r_{i,j}(t)$ denote the amount of remaining execution of $\tau_{i,j}$ at time t. If $\tau_{i,j}$ executes all the remaining execution at speed s, the remaining execution time is $r_{i,j}(t)/s$. The laxity of a job is defined as the maximum amount of time for which the job can idle (or may not execute)without missing its deadline. This amount of time depends upon the speed at which the job executes during its remaining execution. Suppose $\tau_{i,j}$ executes its remaining execution at speed s, its laxity at time t is as the equation below.

$$\boldsymbol{l_{i,j}(t,s) = d - t - \frac{r_{ij}(t)}{s}} \quad (1)$$

3. ENERGY-AWARE EDZL SCHEDULING

3.1 DVFS for EDZL Scheduling

There are some research works on the schedulability of EDZL. Park et al. [18] proposed a utilization-based schedulability test and proved that EDZL dominates EDF, i.e., EDZL can successfully schedule any EDF-schedulable task set. Baker et al. [19], [20] presented an EDZL schedulability test. They demonstrated that the test is superior to existing EDF schedulability tests. Lee et al. [21] proposed another EDZL schedulability test (Theorem 1). They showed that their test outperforms Baker's test with respect to the number of task sets that pass the test.

Theorem 1.(Theorem 2 in [21]) On m-core platforms, a task set τ is schedulable by EDZL if there exists m^*(= 1, 2, …, m) satisfying (2), where $T_1 = \{\tau_i \in \tau \mid \tau_i \notin (m - m^*)$ task with the largest u_i }.

$$\sum_{\tau_i \in T_1} u_i \leq m^* - (m^* - 1) \cdot max\langle u_j | \tau_j \in T_1 \rangle \quad (2)$$

Note that $T_1 = \tau - T_2$ where T_2 is a set of $(m - m^*)$ tasks whose utilization is highest. Briefly, T_1 is a set of $(n - m + m^*)$ tasks whose utilization is lowest.

The first study on DVFS scheme for EDZL scheduling is [14]. Based on Baker's test, Piao et al. [14] presented a technique that calculates a uniform speed on full-chip DVFS platforms. The execution speed of all cores can be safely altered to the uniform speed to reduce energy consumption of cores. Han et al. [22] presented an power-ware EDZL scheduling technique utilizing Lee's test [21]. This technique is simpler and more effective than Piao's because Lee's test is simpler but tighter than Baker's test. On full-chip DVFS platforms where all cores share one clock, the technique computes static uniform speed - all cores operate uniformly at the computed speed. Theorem 2shows that, for a given task set, there exists a uniform speed at which the task set can be successfully scheduled by EDZL on m processing cores.

Theorem 2.(Theorem 1 in [22]) Suppose a task set $\tau$is scheduled by EDZL on m-core platforms. If there exists m^* that satisfies (2), then τ is schedulable with a uniform speed S ($0 < S \leq 1$) that satisfies the following equation.

$$S \geq max\left\{U_{max(\tau)}, \frac{1}{m^*}\left(\sum_{\tau_i \in T_1} u_i + (m^* - 1) \cdot max\{u_j | \tau_j \in T_1\}\right)\right\} \quad (3)$$

On per-core DVFS platforms where each core contains individual clock and its speed can be adjusted individually, the technique computes static individual speed for each task, denoted by $S_1, S_2, \ldots, S_n$, for EDZL scheduling. On this platform, when a job is about to execute on a core, the core's speed is adjusted to the individual speed of the task of the job.Algorithm 1 finds such m^*that minimizes the uniform speed for T_1 and determines the individual speed of each task.

Algorithm 1.(Algorithm 2 in [22]) Calculate individual speed$S_1, S_2, \ldots, S_n$

function ***calculate_individual speed(m, τ)***
1 $S_{min} = 1$, $m_{min^*} = m$
2 **for** m^* **from** 1 **to** m **do**
3 $T_1 \leftarrow \{\tau_i \in \tau | \tau_i \notin m - m^*$*tasks with the largest* $u_i\}$
4 **if** $\Sigma_{\tau i \in T1} u_i \leq m^* - (m^* - 1) \cdot U_{max}(T_1)$ **then**
7 $s \leftarrow$ *calculate_uniform_speed*(m, T_1)
8 **if** $s < S_{min}$ **then**
9 $S_{min} \leftarrow s$, $m_{min^*} \leftarrow m^*$
10 **fi**
11 **fi**
12 **done**
13 $T_1 \leftarrow \{\tau_i \in \tau | \tau_i \notin m - m_{min^*}$*tasks with the largest* $u_i\}$
14 **for** $\tau_i \in T_1$ **do** $S_i \leftarrow S_{min}$ **done**
15 **for** $\tau_i \notin \tau - T_1$ **do** $S_i \leftarrow u_i$ **done**
16 **return** $S_1, S_2, \ldots, S_n$

3.2Combining with MOTE Technique

Algorithm 1 computes individual speed of tasks off-line. We combine the off-line scheme with an on-line technique MOTE [1]. MOTE is integrated with a scheduler. When a job $\tau_{i,j}$ is to be executed on a core, the core's speedis set to the individual speed or s_i^{MOTE} computed by (4).At time t, the scheduler calculates t_{next} by Algorithm 2 of [1], and then computes s_i^{MOTE} by applying Algorithm 3 of [1].

$$s_i^{MOTE} = \frac{w_i^{s_i}(t) \cdot s_i}{min\{d_{i,j}, t_{next}\} - t} \quad (4)$$

Where $w_i^{si}(t)$ is the amount of remaining execution in the worst case assuming the execution speed of $\tau_{i,j}$ is s_i. The core speed is set to $min\{s_i^{MOTE}, s_i\}$.Of course, the core speed should not be lower than s_{min}.

4. EXPERIMENTS

We evaluate the proposed scheme through a simulation. We generate periodic task sets increasing the total utilization of task sets U from $0.25m$ to $0.90m$ with a step of 0.2, where m = 4,8,16.For each U,a group of tasks set is generated. Each group consists of 100 task sets. When we generate a task τ_i, we randomly choose p_i and u_i from a uniform distribution over (10,1000] and (0.1,1], respectively. e_i is given by $p_i \times u_i$. We test every task set using both Baker's test [20] and Lee's [21]. A task set that does not pass either of one is discarded. During simulation, when a job is released, its actual execution time is randomly chosen from [1, e_i].

The processor model of this simulation is Strong ARM SA-1100 processor whose characteristics are given in Table 1. Frequency and supply voltage are altered together to one of the levels. When a core executes a job with speed s, the core's frequency/voltage level is altered to the lowest level whose speed is no lower than s. When a job is completed or preempted, its energy consumption is calculated and summed up. Let a job execute for e time units at a certain frequency/voltage level. Then the amount of energy consumption is $(e/s)P$ where s is the speed and P is the power of the level.

Table 1: StrongARM SA-1100 processor characteristics

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

We compute and sum up the amount of energy consumed by every task in a task set for its hyper-period. Then we normalize it to the amount of energy consumption without any DVFS technique. For each total utilization value, the normalized energy consumptions of task sets are averaged. Figure 1, 2, 3, and 4 demonstrate the average normalized energy consumption for m = 2, 4,8, 16, respectively

As shown in the figures, our combined scheme can further reduce energy consumption. Han' scheme cannot reclaim dynamic slack time that occurs when a job actually demand a less amount of execution than the worst-case. By combining with MOTE our scheme can reclaim such slack time. For instance, when m = 2 and the total utilization is 1.6, our scheme saves 7.92%of normalized energy in average. For a fixed number of cores, on the whole, more energy can be saved as the total utilization increases. For task sets with high total utilization, it is likely that there exist heavy execution tasks. Such tasks may have much slack time if their jobs actually demand far less execution than the worst-case. Those slack time can be reclaimed by on-line DVFS techniques such as MOTE.

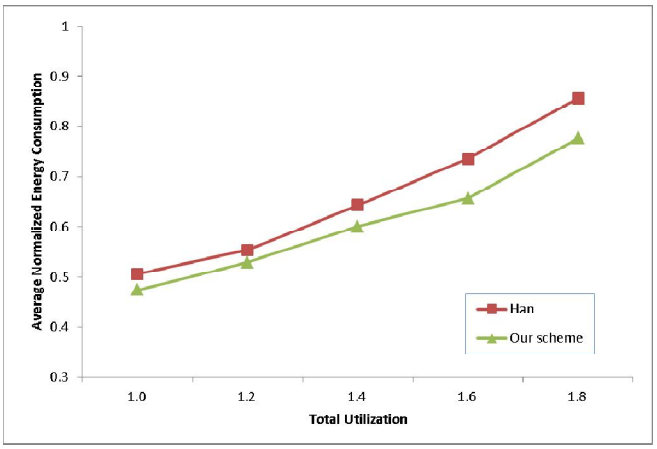

Figure 1: Average normalized energy consumption when m = 2

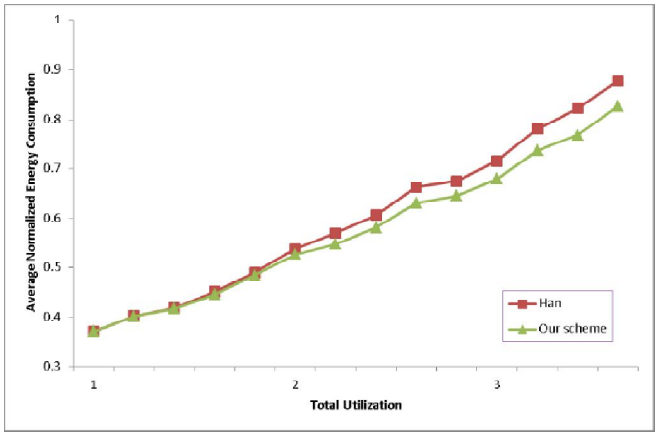

Figure 2: Average normalized energy consumption when m = 4

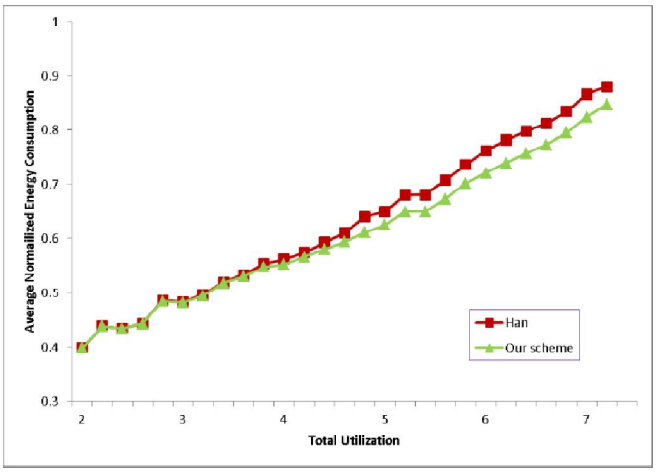

Figure 3: Average normalized energy consumption when m = 8

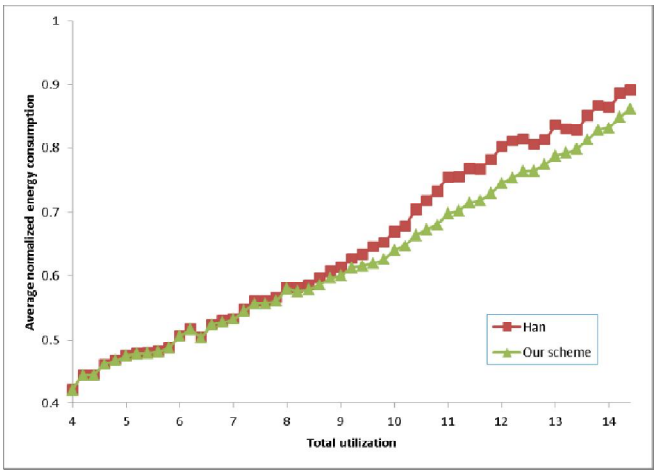

Figure 4: Average normalized energy consumption when $m = 16$

5. CONCLUSION

This paper proposes a dynamic voltage/frequency scaling scheme for EDZL, a global real-time scheduling algorithm. Our scheme combines an off-line DVFS technique (Han's individual task speed scheme) on per-core DVFS platforms with an on-line slack reclamation technique (MOTE) to decrease processing cores' energy consumption. The proposed scheme can save more energy than Han's scheme by at most 7.92%.

ACKNOWLEDGEMENT

This paper was supported by Konkuk University in 2018.

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. https://doi.org/10.1109/4.126534
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. https://doi.org/10.1109/FTFC.2012.6231734
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. https://doi.org/10.1109/RTCSA.2007.37
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. https://doi.org/10.1145/321738.321743
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. https://doi.org/10.1109/SIES.2008.4577677
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. https://doi.org/10.1145/2392987.2392996
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. https://doi.org/10.1007/s11227-014-1310-y
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. https://doi.org/10.1109/ECRTS.2007.14
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. https://doi.org/10.36478/jeasci.2019.8039.8046
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. https://doi.org/10.30534/ijeter/2020/03812020
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. https://doi.org/10.30534/ijeter/2019/04792019
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. https://doi.org/10.30534/ijeter/2019/487112019