Minimizing Energy Consumption of a Chip Multiprocessor Through Simultaneous Core Consolidation and DVFS

Mohammad Ghasemazar, Ehsan Pakbaznia, and Massoud Pedram
University of Southern California
Department of Electrical Engineering
{ghasemaz, pakbazni, pedram} @usc.edu

ABSTRACT **- This paper addresses the problem of minimizing the total energy consumption of a (chip) multiprocessor system while maintaining a required throughput. The minimum energy solution subject to a throughput constraint is achieved by selectively turning cores ON or OFF, assigning a given set of tasks to different cores, and simultaneously selecting the optimum operating supply voltage and clock frequency level for each processor core in the system. This NP-hard problem is solved by a three-level hierarchical framework comprised of a control theory-based dynamic power manager (DPM) and a task assignment unit. Experimental results demonstrate 17% energy saving of the proposed solution approach.**

1. INTRODUCTION

In recent years, with the increase in demand for high performance processors, chip multiprocessor architectures are turned to be a way of maintaining performance scaling despite of the technology scaling problems [1]. At the same time the demand for higher processing power is causing the need for power and energy efficient design of multi-core processing platforms.

Most of the prior research in this field have focused on dynamic power management through core-level responses [2][3][4] or through global task scheduling techniques [5][6][7]. However, the problem of online minimizing the power/energy consumption of a general-purpose Chip Multi Processor (CMP) system while meeting a target throughput by means of core consolidation and DVFS still has potential to improve. DVFS has proven to be an effective approach in reducing energy consumption of single processors [3][8][3][9]. DVFS can be applied to CMP architectures either to all cores (a.k.a. chip-wide) as in [10], or to individual cores independently (a.k.a. per-core) as in [11][12]. In [10], the authors address the problem of finding a chip-wide operating voltage-frequency (*v-f*) setting as well as finding the number of ON cores that minimize CMP power under a performance constraint, using a priori application characterization. In [2], the authors deploy a proportional-integrator (PI) controller to perform DVFS in CMPs. This work does not take advantage of the large potential power saving of changing the number of ON cores. Similarly, in [11], the authors introduce the concept of a global power manager which provides a hierarchical mechanism to set the optimum DVFS setting of each core while meeting a target power budget. However, the number of the ON cores is always fixed independent of the workload.

In this paper, we address the problem of minimizing the total energy consumption of a CMP while maintaining a required throughput. The minimum energy solution subject to a throughput constraint is achieved by selecting and applying the optimum operating supply voltage and clock frequency level for each processor in the system as well as turning them on or off while partitioning and assigning a given set of tasks to different processors. The problem is mathematically formulated and then solved by using a hierarchical algorithm. The main contributions of this work can be summarized as: introducing an on-line global power manager that performs core consolidation and DVFS while solving task assignment problem in CMP platform.

2. SYSTEM MODEL

In this study, we consider an M-way homogeneous *Chip Multi-Processor* (CMP) system which is a multicore system consisting of M identical processing cores. The cores are assumed to be independent, except that they share the L2 cache and interface to the main memory [13]. The cores are capable of independently running at different voltage-frequency (*v-f*) settings. Figure 1 shows our CMP configuration. The *Power Management Unit* (PMU) monitors working status of all blocks and CMP throughput, and determines the optimal number of ON cores and their *v-f* settings. Incoming tasks are held in the *Global Queue*, GQ, before they are assigned to the *Local Queues*, LQ's of individual cores by the Task Assignment Unit.

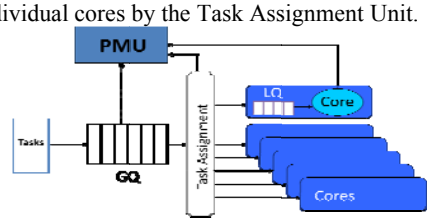

Figure 1. System model with global and local queues.

Programs and applications running on the system and the operating system generate tasks and send them to CMP hardware. Similar to [14], we assume these tasks are independent. (The problem of optimally assigning dependent tasks in multiprocessor systems has been widely studied in the literature [6], but it is outside the scope of the present paper.) The task characteristics used in this paper are the expected *execution time*, τ, and expected *instruction per cycle* (IPC) values. If these characteristics are precisely known *a priori*, an optimal solution to the problem would exist. But, this is not practical in most cases, because we cannot precisely determine IPC and execution time before executing the tasks on the target hardware in any particular operating condition. Task characteristics can be determined and attached to the task by the application software, or can be estimated adaptively by hardware at runtime. For any certain application, it can be assumed that the behavior of future tasks is similar to what has been observed in the recent past tasks. Maintaining a history is practical by means of on-chip power and performance sensors widely available in state-of-the-art processors [13]. However, some error and uncertainty in the estimated values is inevitable and should be taken care of. In this paper, the uncertainty and inaccuracy about task characteristics will be taken care of by using a feedback control loop (c.f. section 4.2.14.2).

3. MINIMUM ENERGY CMP DESIGN PROBLEM

Minimum Energy CMP Design Problem (MECD) is an optimization problem which minimizes the *energy* consumption of a CMP given a *throughput* constraint. The optimization is done through optimally adjusting number of ON cores and their *v-f* settings through DVFS, as well as finding the optimum task assignment to ON cores. Note that in this work, we do not consider any thermal optimization or constraint.

3.1 ENERGY AND THROUGHPUT MODEL

Energy dissipation of cores in a CMP in a time interval, T_d, can be defined as the summation of the energy consumption (idle plus active) of each of the cores. That is:

$$E_{T_d} = \sum_{i=1}^{M} P_{idle_i} \times t_{idle_i} + \sum_{i=1}^{M} P_{active_i} \times t_{active_i} \qquad (1)$$

where M denotes the number of cores, and P_{idle_i} and P_{active_i} are idle and active power consumption values for the i^{th} core, respectively. t_{idle_i} and t_{active_i} are the idle and active time of the i^{th} core in the time

period, T_d. An ON core is active if it is executing a task; otherwise, it is in the idle mode ($t_{idle_i} + t_{active_i} = T_d$). For an off core, both active and idle times are zero.

Throughput of a CMP during time period T_d is defined as the average number of the executed instructions per unit of time. If a core is executing task *k,* at frequency f_j, its throughput is given as:

$$IPS(f_j) = IPC_{kj} \cdot f_j \quad (2)$$

where IPC_{kl} is represents the average instruction per cycle of task *k* when running under the *v-f* settling level *j*. It depends on processor micro architecture (e.g. branch prediction mechanism, execution resources, cache structure, etc.), nature of task (e.g. number of branches and memory accesses in the task) and the clock cycle time. Using equation (2), the CMP throughput can be calculated as the summation of throughput of individual cores:

$$IPS_{CMP} = \frac{1}{T_d}\sum_{i=1}^{M}\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ij}a_{ki}IPC_{kj}f_j\tau_{kj} \quad (3)$$

where, x_{ij} is binary assignment variable of core *i* to *v-f* setting level *j* (x_{ij}=1 means that i^{th} core is using the j^{th} *v-f* level,) a_{ki} indicates whether task *k* is assigned on core *i* or not, and $\tau_{k,j}$ is the expected execution time of task *k* when it is running on a core with the j^{th} *v-f* setting; τ_{kj} is a given characteristic of task *k*.

3.2 Problem Formulation

MECD may be formulated as minimizing CMP energy dissipation in a time interval T_d, under an average throughput constraint,:

$$Minimize\{E_{T_d}\}$$

subject to:

$$\frac{1}{T_d}\sum_{i=1}^{M}\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ij}a_{ki}IPC_{kj}f_j\tau_{kj} \geq IPS_{req} \quad (4)$$
$$x_{ij} \in \{0,1\}, \quad \forall i : \sum_{j=1}^{J} x_{ij} = 1$$
$$a_{ki} \in \{0,1\}, \quad \forall k : \sum_{i=1}^{M} a_{ki} \leq 1$$

in which a_{ki} and x_{ij} are variables of optimization. The first constraint is the system level throughput requirement, IPS_{req}. The second constraint ensures that exactly one (v_j, f_j) setting for each core is used, while the last one ensures that every task is assigned to at most one core. A restricted form of MECD problem is equivalent to multi-processor scheduling problem, that its complexity is known to be NP-hard [5]. Furthermore, due to the existent uncertainty of the task characteristics (execution time and IPC) and the real time nature of the solution, conventional mathematical optimization approaches do not result in a robust solution to this problem. We want to utilize an efficient (light and thin) and robust algorithm to solve this problem.

4. Proposed Solution

In the CMP framework described in the previous section, there are three power minimization enablers namely number of ON cores, DVFS, and task assignment. We solve MECD problem by using a multi-tier power management strategy that decides on these three mechanisms, at three hierarchical levels with different timing granularities. A logical order is to first determine the number of ON cores and their operating frequency, and then find a feasible assignment of tasks to the ON cores. In the current technology, energy and performance overheads of changing the *v-f* setting are much lower than those of turning on/off a core and required task migration. Therefore, in our algorithm we first solve the core consolidation problem, and then based on the solution found, adjust DVFS setting of cores to match the required throughput at a lower hierarchical level. Finally, the task assignment is done according to the *v-f* setting of cores and task characteristics to establish a feasible task allocation. Our algorithm determines the number of ON cores periodically with a fixed period of T_d, *decision period*, and performs DVFS and task assignment with periods T_s, *sampling period*, and T_a, *assignment period*, respectively. We have: $T_d > T_s > T_a$.

Due to inherent uncertainty and variability of task characteristics, an open-loop solution suffers either from overestimating or underestimating the frequency values, which in turn results in excessive energy consumption or throughput constraint violation. Therefore, a feedback-based control method is exploited in this paper, to dynamically adjust the *v-f* settings of individual cores. In this work we assume that since all the cores are exactly similar and the tasks characteristics are uniformly distributed over working range, a single chip-wide frequency is applied to all cores, as suggested by [10]. Therefore the feedback loop will determine a single optimum frequency for all the cores. For this, a continuous range of the frequency levels and the corresponding voltage level is needed, which is not practical in the current technology. Instead, modern DVFS-enabled circuits employ several pre-determined frequency and voltage levels. A quantization block transforms continues frequency value to a set of available discrete frequencies.

Figure 2 shows the top level block diagram of our proposed solution. This DPM can be implemented partially in hardware and partially in software (as a high-priority task being executed on one of the cores.)

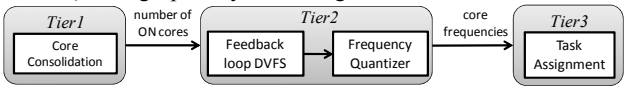

Figure 2. Block diagram of the proposed solution.

4.1 Determining the Number of ON Cores

Let's denote the number of the ON cores in the CMP by *m*. A given throughput value can be achieved by several solutions for *m* (with *v-f* adjusted correspondingly), each one having different energy dissipation value (c.f. Figure 6). Our approach for finding the minimum energy *m* value, m_{opt}, is to do a steepest descent search. At the beginning of each decision epoch (every T_d), a new *m* value is being calculated. We calculate energy values for the adjacent points of the current *m* (*m*+1 and *m*-1) based on the model described in section 3.1 and find the direction in which energy decreases. We then update the value of *m* and continue the search toward the direction that leads to lower energy consumption, until we find the optimum value for *m*; then we fix this number for a time interval of T_d and seek optimal *v-f*.

4.2 DVFS Using Feedback Control Loop

Given the optimum *m*, a closed loop controller determines and updates a single optimum frequency for all the cores, by measuring the error in the actual CMP throughput, *H*, compared to the target throughput, H_{target}, as shown in Figure 3. It is done periodically with period T_s. To use the linear control theory, we linearize CMP throughput, *H*:

$$H(f) = IPS = m \cdot \overline{IPC}_{max} \cdot f \quad (5)$$

where $\overline{IPC}_{max}$ is the maximum value of average core *IPC* for all tasks. Considering its maximum value ensures stability of controller.

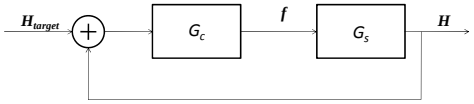

Figure 3. Closed-loop controller system representation.

4.2.1 Closed-Loop Controller to Set v-f Settings

In this paper, we use Proportional Integral (PI) controller for its superior performance for linear systems [15]. We follow the well-established control theory techniques to obtain the characteristic equation of the closed loop system (c.f. equation (6)) and stabilizing the controller through appropriate pole placement to meet desired controller characteristics such as response time and overshoot percentage [15]. The system's characteristic equation in z-domain is:

$$z^2 + \left((K_I + K_P) \cdot \overline{IPC}_{max} - 2\right)z + \left(1 - K_P \cdot \overline{IPC}_{max}\right) = 0 \quad (6)$$

where K_p and K_I are coefficients to be determined for each system configuration (i.e. *m* value) for a better accuracy and stability.

4.2.2 Frequency Quantization

As explained earlier, a quantization block transforms the calculated optimum frequency value to a pair of available frequencies. If the optimum frequency given by controller, f_{opt}, lies between two available physical frequency levels $f_1 \leq f_{opt} < f_2$, we set the frequency of each core to either f_1 or f_2 such that throughput is met and energy is minimized. This is achieved by solving the following problem and finding f_1 and f_2.

$$Maximize\ \{p\}$$

subject to:

$$p \cdot H(f_1) + (m-p) \cdot H(f_2) \geq IPS_{req} \quad (7)$$

where p is the number of cores running at f_1.

4.3 ASSIGNING TASKS TO ON CORES

In this section we determine whether each task is executed on a core with frequency f_1 or f_2 (the two optimum frequencies found in the Section 4.2.2.) Suppose the total workload that must be assigned to various cores of a CMP is W at a known reference frequency of f_0 (the total time required to execute all the tasks on a core at frequency f_0). Consider that we divide W in two parts, W_1 and W_2, where W_1 is the part of the workload that will be executed on cores with frequency f_1 and W_2 is the remaining workload that will be executed on cores with frequency f_2 ($f_2 > f_1$). Clearly, the actual execution time is proportional to the ration of operating frequency (e.g. f_1) to f_0. The energy consumption for running the total workload, is calculated as follows:

$$E_{tot} = (L_1 + L_2)T_a - \left(\frac{L_1}{f_1}W_1 + \frac{L_2}{f_2}W_2\right)f_0 + \left(\frac{A_1}{f_1}W_1 + \frac{A_2}{f_2}W_2\right)f_0 \quad (8)$$

where L_1 and L_2 are the idle power consumption values corresponding to f_1 and f_2, respectively, and A_1 and A_2 are the active power consumption values corresponding to f_1 and f_2, respectively.

Lemma 1: The minimum total energy consumption is achieved when W_1 is maximized, as long as the throughput constraint is satisfied. (Proof is omitted due to space limit.)

Using Lemma 1, we first assign as many tasks as possible to cores with frequency f_1, and the rest of the tasks to the cores with frequency f_2. To do so, we start with the first core with frequency f_1 and we allocate the maximum possible workload on this core. Next, we continue this until we finish with all cores running on f_1. Then, we continue by allocating the remaining tasks, if there is any, to the cores with working frequency f_2. This can be done in polynomial time using *dynamic programming*. More precisely, to assign the maximum workload on each core, we are given K items, tasks, where the k^{th} item has a nonnegative weight that is τ_{kj}, the execution time of the k^{th} task on frequency f_j. We are also given a bound Γ which is the available service time for the core under the consideration. We would like to select a subset S of the tasks so that $\sum_{k \in S} \tau_{k,1} \leq \Gamma$ and, subject to this constraint, $\sum_{k \in S} \tau_{k,1}$ is maximized. This is the so called *Subset Sum Problem (SSP)* which is a special case of the *Knapsack* problem and can be solved in polynomial time using dynamic programming [14].

The proposed heuristic algorithm can be summarized as:

1. *Pick the initial number of ON cores, m*
2. *Search for optimum m (see section 4.1) {*
3. *Find the optimum frequency level for all ON cores by using the closed-loop controller (see section 4.2)*
4. *Find the optimum number of cores running at f_1 or f_2 (see section 4.2.2)*
5. *Predict the energy consumption for current m }*
6. *Determine f1 and f2 at each sampling time (see section 4.2)*
7. *Perform Task Assignment to cores given their v-f settings (see section 4.3)*

The worst-case complexity of the proposed heuristic is $O(M \times logM)$ since the loop of lines 2-5 is repeated at most M times, and finding optimum f_1 and f_2 by means of described search methodology is done in $O(logM)$.

5. EXPERIMENTAL RESULTS

We have developed a real-time simulator in C++ to implement and evaluate the proposed power management technique. This simulator is not a cycle-accurate simulator actually running the tasks; instead, it emulates execution of the tasks on cores. It is an event-driven simulator, in which the triggering events are task arrival, task departure, decision point, sampling point, and assignment point. GQ, all the LQ's, PMU, and task assignment are implemented in our emulator. We simulated 4, 6, and 8-way CMPs whose core configuration is as described in Table 1 which is based on the configuration of alpha-21264 processor [16].

Table 1. Configurations of the cores in CMP system

Fetch/issue/commit	4/4/4
Functional Units	4INT ALU, 2INT Mul/Div,4 FP ALU,2 FP Mul/Div
Load/Store queue	32/32
L1 instruction/ data cache	64KB, 64B block, 2way, 1-cycle / 3-cycle latency (at f_{max}), LRU replacement
L2 unified cache	2N MB, 64B block, 1-way, 14-cycle lat, LRU

Table 2. Simulation parameters

Parameter	Value
Core Freq. Levels	{Off, 600:100+:2600MHz}
LQ / GQ	8 / 40,80,160
L1/L2 cache	64KB/ {4, 8, 16 MB}
T_d / T_s / T_a	50 / 10 / 2ms
E(τ)	0.5ms
E(λ)	0.6N, N, 1.6N [1/ms]
No. of simulated tasks	5000

The expected execution time and inter-arrival time of tasks are assumed to be two independent random variables with exponential distributions with mean values E(τ) and 1/E(λ), respectively. Where, λ is the task arrival rate. Note that in order to avoid overflow, the average of task arrival rate must be less than or equal to the maximum processing capacity of CMP. Table 2 summarizes the parameters used in the simulation. For tasks generation, we choose a parent application from nine SPEC2000 benchmarks [17] with equal probability, p=1/9. In order to model the uncertainty of the information about the tasks, we apply ±20% uniform disturbance to execution time, and IPC of each task at the runtime.

The baseline for comparing the proposed method is the exact solution to the mathematical problem in (4) presented in section 3.2. The exact solution is obtained by doing an exhaustive search among all possible *v-f* settings for each core and all task assignment combinations. As expected, the search space grows exponentially in size with the number of tasks, K, the number of cores, M, and the number of available *v-f* levels, J. Counting only the valid combinations, there are a total number of J^M possible configurations in this multi-core system. Task assignment can also be done in M^K ways. As a result an exhaustive search must test $J^M.M^K$ different cases. For small J, M, and K values, we can find globally optimum solution by enumeration.

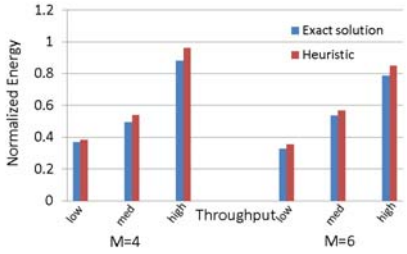

Figure 4. Energy consumption of exact solution vs heuristic

Figure 4 shows the normalized total energy consumption of the system for the exact solution which leads to the optimum solution of the MECD problem and the proposed heuristic (normalized to its maximum value.) The experiment has been done for two different CMP's with M=4 and M=6 cores, and for three throughput constraints for each configuration (low, medium and high corresponding to 2000, 5000, and 10000 MIPS, respectively). The reason that the energy consumption of the system with six cores is smaller is due to the fact that the required IPS's were the same for both systems, thus the system with higher number of available cores can run at lower frequencies on average, resulting in less power dissipation.

Results of this figure show that total energy consumption of our solution is within 9% of the oracle solution which solves the mathematical formulation in (4) having all the detailed information about the tasks whose complexity is exponential. The actual simulation time of the two methods differ by two orders of magnitude.

To assess the efficacy of the feedback controller in our proposed solution, we compare it to one without the feedback loop (the so-called open-loop control where the frequency is calculated based on given performance constraint without any feedback from actual

system measurements). Figure 5 shows the effectiveness of the closed-loop controller for high and low performance constraints. It can be seen that the closed-loop controller achieves higher energy saving of about 10% at high and low workloads. This is because it can adjust the *v-f* setting of the cores to follow the required performance level more closely, thereby avoiding excessive energy consumption.

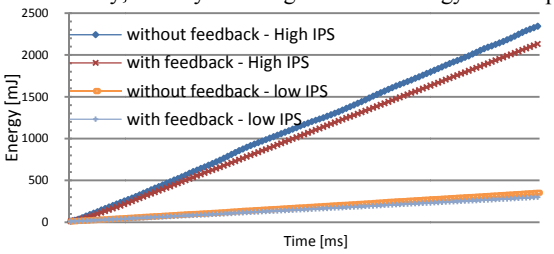

Figure 5. Energy consumption with and without feedback

Figure 6 shows importance of shutting off the unused cores. At a decision point, with a given throughput, increasing the number of cores always results in decrease in the optimum frequency, but not necessarily a decrease in the energy consumption. It can be seen that the case m=3 has a lower energy consumption than m=4.

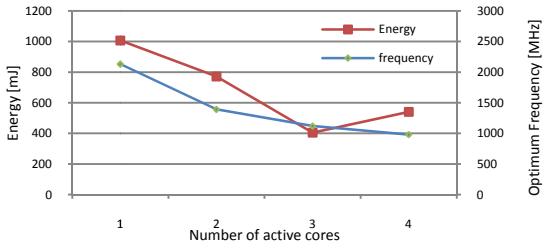

Figure 6. Energy consumption of different number of ON cores

Finally, to perform an overall evaluation of our proposed PM technique, we compare it with a relatively energy efficient baseline PM. It implements same power reduction techniques as our method, except that it utilizes open loop DVFS, and it does not support core consolidation. It calculates the frequency as the reverse function of (2), and adds 10% to the calculated value in order to satisfy the throughput constraint in existence of uncertainties. Figure 7 compares the frequency setting and throughput of our proposed algorithm to this baseline. The baseline PM always runs with all cores ON, and their frequency calculated as described. The closed loop frequency is always lower for same throughput constraint. Moreover, in the second 100ms, the baseline chooses lower frequency with m=4 cores running, while our proposed method uses m=3 cores. The proposed PM gains an overall energy consumption of **17%**.

6. CONCLUSION

We formulated the problem of minimizing the total energy consumption of a (chip) multiprocessor under a throughput constraint. We presented a mathematical formulation for the problem in the form of a nonlinear mixed integer programming problem and discussed its solution. An algorithm was proposed to solve the same problem efficiently. It is basically a three level hierarchical approach which includes a global dynamic power manager (DPM) and a task assignment unit. The global DPM is at a higher level in the hierarchical system; it adjusts the voltage and frequency settings of cores considering the workload and incoming task rate. Then, the task assignment unit determines which subset of tasks runs on each core with known state. Comparison of the experimental results of to proved the high quality of the proposed algorithm (about 17% saving in the energy consumption).

REFERENCES

[1] M. Pedram, J. M. Rabaey, *Power Aware Design Methodologies*, Springer, 2002

[2] S. Herbert, D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," *Proc. ISLPED,* 2007.

[3] F. Xia, *et al.*, "Control Theoretic Dynamic Voltage Scaling for Embedded Controllers," *IET Computers and Digital Techniques* 08.

[4] R. Rao and S. Vrudhula, "Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors," *Proc. ICCAD* 2008.

[5] H. Aydin, Q. Yang, "Energy-Aware Partitioning for Multiprocessor Real-Time Systems," *Proc. IPDPS*, 2003.

[6] Y. Xie , W. Wolf, "Allocation and scheduling of conditional task graph in hardware/software co-synthesis," *Proc. DATE,* 2001.

[7] M. Harchol-Balter, *et al.*, "On choosing a task assignment policy for distributed server system," *IEEE Journal of Parallel and Distributed Computing*, vol59, 1999.

[8] P. Juang, *et al.*, "Coordinated, distributed, formal energy management of chip multiprocessors," *Proc. of ISLPED,* 05.

[9] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration, " *Proc. ISCA,* 06.

[10] J. Li, J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," *Proc. HPCA,* 2006.

[11] C. Isci, *et al.*, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," *Proc. IEEE/ACM MICRO,* 2006.

[12] M. Ghasemazar, *et al.* , "Minimizing the Power Consumption of a Chip Multiprocessor under an Average Throughput Constraint," *ISQED,* 2010.

[13] http://www.intel.com/products/processor_number/chart/xeon.htm

[14] J. Kleinberg, E. Tardos, *Algorithm Design*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2005.

[15] R. C. Dorf, R. H. Bishop, *Modern Control Systems*, Prentice Hall, 2008.

[16] R. E. Kessler, "The Alpha 21264 Microprocessor," IEEE MICRO, 1999.

[17] http://www.spec.org/

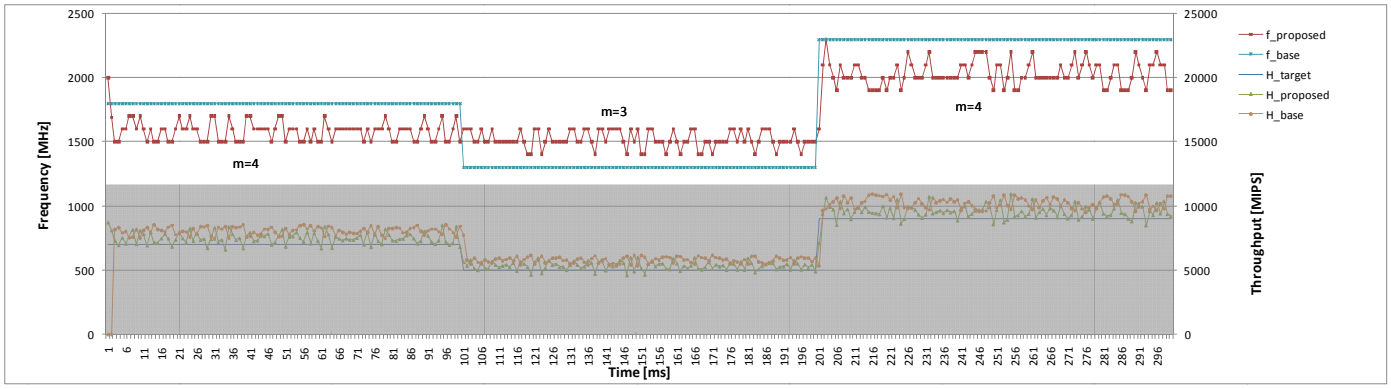

Figure 7. Throughput and working frequency comparison of the proposed algorithm vs. baseline, resulted in 17% energy saving