Temperature-Aware Task Mapping for Energy Optimization with Dynamic Voltage Scaling

M. Bao, A. Andrei, P. Eles, Z. Peng
Embedded Systems Laboratory
Department of Computer and Information Science
Linkoping University, Sweden, SE-581 83
Email: minba, alean, petel, zebpe@ida.liu.se

***Abstract*—Temperature has become an important issue in nowadays MPSoCs design due to the ever increasing power densities and huge energy consumption. This paper proposes a temperature-aware task mapping technique for energy optimization in systems with dynamic voltage selection capability. It evaluates the efficiency of this technique, based on the analysis of the factors that can influence the potential gains that can be expected from such a technique, compared to a task mapping approach that ignores temperature.**

I. Introduction

In multi-processor platform, task mapping, first addressed by Chu et al. [3], is the process of allocating computational tasks and data transfers to processing elements (PEs) and communication links (CLs) [16]. Known as a NP complete problem [6], task mapping in distributed system has very often been solved using efficient heuristics that can be applied to large scale applications.

Due to increasing demands on performance, embedded applications are frequently implemented on multiprocessor systems on chip (MPSoC). Very often they are required to satisfy strict timing constraints and are functioning with a limited energy budget. One of the preferred approaches for reducing the overall energy consumption is dynamic voltage selection (DVS). This technique exploits the available slack times by reducing the voltage and frequency at which the processors operate and, thus, achieves energy efficiency. In the context of system-level design, energy optimization should be achieved by jointly conducting task mapping, scheduling and DVS in the design flow. Some proposed algorithms [16], [21] do mapping, scheduling and DVS in successive steps and sequentially conduct each step in every iteration of the optimization loop, others, oppositely, make simultaneous decisions for mapping, scheduling and DVS [8], [14].

The high power densities achieved in current system of chip (SoCs) do not only result in huge energy consumption but also lead to increased chip temperatures. High temperatures can impact reliability as well as cooling and package cost. Based on the development of temperature modeling and analysis tools e.g. [5], [20], several temperature-aware system level design approaches have emerged. Wang et al. [18] proposed an approach to task scheduling under peak temperature constraints. Design space exploration for MPSoCs architectures under area and thermal constraints is presented by Li et al. [9], while in [15] Sankaranarayanan et al. advocate thermal aware floor-planning. DVS, as an important energy optimization technique, has also been extended to be temperature-aware in our previous work [2]. For task mapping, Xie et al. [19] are the first to consider the temperature issue but their algorithm does not use DVS for energy optimization.

The objective of this paper is to propose a new temperature-aware task mapping approach with the goal of energy optimization. A genetic algorithm for task mapping is introduced which is based on a temperature aware DVS technique. We also perform a thorough analysis of some parameters that influence the potential gains that can be expected from a temperature aware task mapping technique, compared to an approach that ignores temperature.

This paper is organized as follows: section II will introduce our system and application model as well as the temperature-aware voltage selection technique. A motivational example is then given in section III. The optimization technique is described in section IV and the experimental results are presented in section V. Finally, conclusions are drawn in section VI.

II. Prelimilries

A. System Model

We consider systems realized as bus based multi-processor architectures on chip. We assume that the processors can operate in several discrete execution modes. An execution mode is characterized by a pair of supply and body bias voltages: (V_{dd}, V_{bs}). Each execution mode has an associated frequency and power consumption (dynamic and leakage). The functionality of the application is captured as a set of task graphs. In a task graph $G(\Pi, \Gamma)$, nodes $\tau \in \Pi$ represent computational tasks, while edges $\eta \in \Gamma$ indicate data dependencies between tasks (communication). For each task, the worse case number of cycles to be executed is given. And each task is annotated with deadlines that have to be met at run-time.

B. Temperature-Aware DVS

In [1] we have presented an approach to combined supply voltage selection and adaptive body biasing. Given a multiprocessor architecture and a mapped and scheduled application, the DVS algorithm in [1] calculates the appropriate execution

modes (V_{dd} and V_{bs}) for each task, such that the total energy consumption is minimized. Another input to the algorithm is the dynamic power profile of the application, which is captured by the average switched capacitance of each task. This information will be used for calculating the dynamic energy consumed by the task in a certain execution mode according to the energy model presented in [1]. Similarly, in [1] the equations are presented which are used to calculate leakage energy, during the optimization process. However, since leakage strongly depends on temperature, an obvious question is which temperature to use for leakage calculations. Ideally, it should be the temperature at which the chip will work when executing the application. This temperature, however, is not known, since the algorithm is just calculating the voltages at which to run the system and these voltages are influencing the energy dissipation which, again, is determining the temperature.

The algorithm in [1] requires the designer to introduce an assumed temperature which is used at energy optimization. This, of course, leads to suboptimal results, since the temperature used for energy calculation during voltage selection is different from the actual temperature at which the chip works. In order to overcome the above problem, in [2] we have proposed a temperature aware DVS technique which is based on an iterative approach as illustrated in Fig. 1.

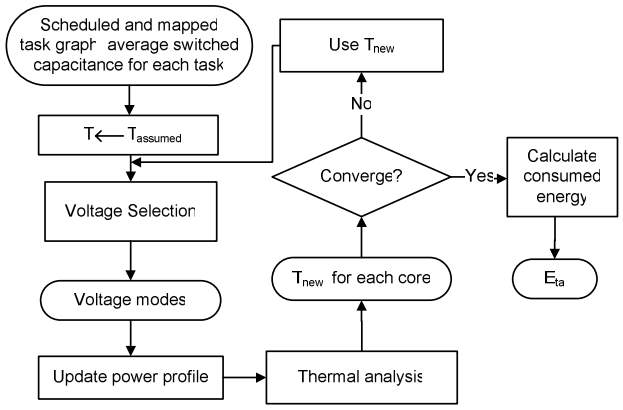

Fig. 1. Temperature-aware Voltage Selection

Given is a task graph mapped and scheduled on a multicore SoC, and the average switched capacitance for each task. A so called "assumed" temperature, at which each core is supposed to run, is also fixed as input. The voltage selection algorithm will determine, for each task, the voltage modes (V_{dd} and V_{bs}) such that energy consumption is minimized. Based on the determined voltage modes (and the switched capacitances known for each task) the dynamic power profiles are calculated and the thermal analysis is performed and the temperature is determined for each core in steady state. This new temperature is now used again for voltage selection and the process is repeated until the temperature converges. Convergence means that the actual temperature values used at voltage selection correspond to the temperature at which the chip will function when running with the calculated voltages.

Thermal analysis in our DVS technique is based on HotSpot [5], our modifications to HotSpot, in order to capture the dependence of leakage on temperature, are presented in [2]. As shown in [2], in most of the cases, convergence is reached in less than 5 iterations.

The above DVS technique assumes that the task graph is already mapped. If this mapping, however is performed before the DVS step, it has to be based on an approach which ignores the temperature at which cores are running. As we will show in the next section, this can result in significant energy losses. Therefore, in the rest of this paper, we will present and evaluate a temperature aware task mapping technique.

III. Motivational Example

In this section we highlight the importance of considering temperature during the task mapping process. Consider an application containing five tasks (Fig. 2). They are going to be mapped on two identical processing elements: PE1, PE2. The application has a global deadline of 0.004s. The workload of the task task1..task5 is (in clock cycles) $3.32 * 10^4$, $2.5 * 10^4$, $3.0 * 10^5$, $3.5 * 10^4$, $5.5 * 10^5$ respectively, and their average switched capacitance is $3.9 * 10^{-8}$, $3.9 * 10^{-8}$, $3.5 * 10^{-8}$, $3.6 * 10^{-8}$, $3.6 * 10^{-8}$ correspondingly.

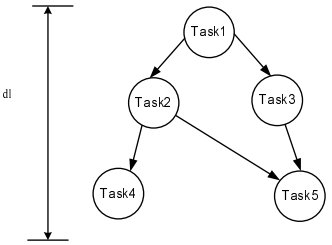

Fig. 2. Motivational Example: Task Graph

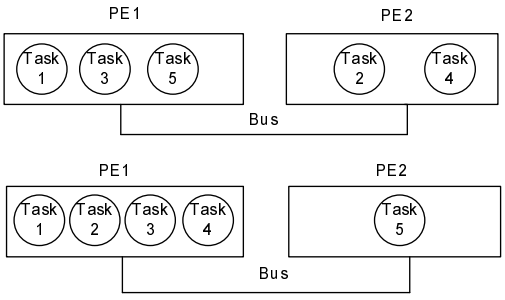

Fig. 3. Motivational Example: Two Task Mapping alternatives

Let us consider the two mapping alternatives in Fig. 3. For each of the two mapping alternatives, the following steps are performed [1]:

[1]For simplicity in this example we do not consider the energy consumed for communication, this energy, however, is calculated in our mapping optimization flow.

1) Construct a task schedule which determines the order of execution.
2) Run the temperature unaware DVS algorithm (the one described in [1]) which will generate the voltage levels for each task. As discussed in section II-B, the DVS algorithm considers, for energy calculation, an "assumed" temperature provided by the designer (in our case 90°C).
3) Calculate the consumed energy for each task and for the whole system, considering that the application runs at the assumed temperature. This energy (denoted E_1 in Table I), differs, of course, from the actual energy dissipated by the application, but it constitutes the best approximation that can be obtained without a temperature aware technique.
4) Using the voltage obtained at step 2, perform the thermal analysis of the system and determine the temperature for each core in steady state.
5) Using the temperature obtained in step 4, calculate the energy consumed by each task and by the whole system. This energy (denoted E_2 in Table I) represents the real dissipated energy.

Table I shows that energy values E_1 and $E_2$of the tasks and the actual steady state temperature of the cores, corresponding to the two mapping alternatives. It is important to notice that a designer using a temperature unaware technique has only access to the energy values E_1. Based on these values, the designer will select the mapping alternative 1 as being more energy efficient, since the value of E1 in this case (0.0347J) is smaller than the one for Mapping2 (0.0351J).

However, if we consider the real energy consumption, E_2, it turns out that, in reality, Mapping2 is more energy efficient and using that mapping, instead of Mapping1, reduces the energy consumption by 22% (0.0263J instead of 0.0338J). What this example shows is that temperature has to be taken into consideration when deciding on an energy efficient mapping of a MPSoC application. Such a temperature aware mapping technique is presented in the following section.

TABLE I
MOTIVATIONAL EXAMPLE

Task	Temperature(°C)	E_1(J)	E_2(J)
Mapping1:			
Task1	PE1:89.7	0.0011	0.0011
Task2	PE2:50.1	0.0008	0.0005
Task3	PE1:89.7	0.0113	0.0113
Task4	PE2:50.1	0.0011	0.0006
Task5	PE1:89.7	0.0204	0.0203
Total		0.0347	0.0338
Mapping2:			
Task1	PE1:58.4	0.0011	0.0008
Task2	PE1:58.4	0.0008	0.0005
Task3	PE1:58.4	0.0113	0.0080
Task4	PE1:58.4	0.0011	0.0007
Task5	PE2:70.2	0.0208	0.0162
Total		0.0351	0.0263

IV. MAPPING HEURISTIC

Our temperature aware mapping approach is based on a genetic algorithm (GA). By imitating and applying the principles of natural selection and "survival of the fittest" on a population pool (consisting of several solution candidates), GAs are able to gradually improve the quality of solutions and to evolve towards close to optimal result [7]. Each solution candidate individual is encoded as a string (chromosome) and is associated with a solution quality (fitness). Based on their fitness, the individuals are ranked within the selection pool. In each iteration, the highest ranked individuals are selected for reproduction by mating (crossover) with other individuals of the population. The produced offspring replaces least ranked solutions in the population. Occasionally, new individuals are also generated by mutation. A mutation is realized by randomly changing the value of certain genes of a chromosome. Fig. 4 illustrates the overall flow of our temperature-aware mapping approach.

The encoding of individual mapping candidate into chromosomes is illustrated in Fig. 5. A chromosome is represented by an array and each element of the array (gene of the chromosome) captures the mapping of a task to a processing element. Thus, in Fig. 5, according to Mapping 1, Task2 is mapped to PE2 and Task4 to PE1.

For our heuristic we use a total population size of 25. The fitness function for each individual is evaluated as the energy consumption corresponding to the respective mapping alternative (smaller energy consumption means higher fitness). In order to compute the energy, the application is scheduled [2] after which our temperature-aware voltage selection presented in [2] is applied, which calculates voltage levels for each task, such that the total energy consumption is minimized (section II-B). Based on the calculated voltage and actual temperatures, the total consumed energy is obtained.

Based on the energy (fitness) value the individuals for mating are selected, using a roulette wheel technique [7]. According to this selection rule, high fitness individuals have a high probability to mate. For mating, multi-point (2 to 8 points) crossover is conducted (Fig. 6). The value of crossover points is determined based on the length of the chromosome (number of tasks). Mutations (Fig. 7) are applied with a probability of 5% on 10% of the chromosome genes (the percentage has been determined experimentally).

The individuals for the next generation are selected out of the pool formed by the old population and the new offsprings. Survival of high fitness (smaller energy) individuals is again favoured.

The exploration process is terminated when a stopping criterion has been reached and the best ever mapping solution is returned. We are terminating the optimization process if for ten successive generations the reduction of energy consumption for the best solution produced is less than 1%.

[2]Task scheduling is performed using a list scheduling based approach described in [4]

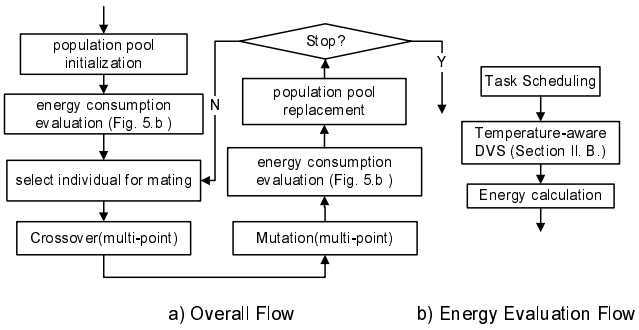

Fig. 4. Temperature aware mapping

V. Experimental Results

Experimental results presented in this section are aimed at exploring the efficiency of being temperature-aware for task mapping, compared to previous mapping algorithms which ignore the temperature issue. For our experiments we have used both randomly generated applications and real-life examples.

We have randomly generated applications consisting of 60 to 400 tasks. The applications are mapped on a MPSoC architecture consisting of 9 identical cores. The cores are running at 10 different supply voltage levels in the range [0.6V, 1.8V]. The temperature model related coefficients are the same as in [2], while the power models and associated parameters are the same as in [2] [12] [11]. The total workload of an application (later referred as TW) is randomly generated in the interval [10^7, $9 * 10^7$] cycles. The size of individual tasks is in the interval [10^3, 10^6] cycles.

We have generated three sets consisting of 50 applications each. For the first set, S_1, the applications are such that 75% of the TW is realized by the tasks with sizes in the interval [10^4, 10^5] cycles, while the rest of the TW is realized by the tasks with sizes in the interval [10^3, 10^4] or [10^5, 10^6] cycles. In the second set S_2, tasks sizes are distributed over the whole interval [10^3, 10^6] cycles. Finally, the set S_3 consists of applications, in which task sizes are either in the small or the large range: only 5% of the TW is realized by tasks with sizes in the middle range [10^4, 10^5] cycles, while the rest of the TW is realized with tasks in the extreme intervals [10^3, 10^4] or [10^5, 10^6] cycles.

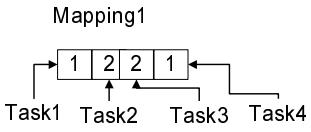

Fig. 5. Encoding

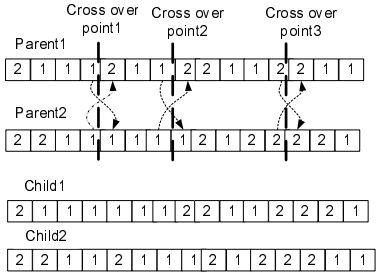

Fig. 6. Multi-point Cross Over

Given a certain application we run the temperature-aware task mapping (later referred as TaTM), as described in section IV (Fig. 4), and obtain the optimized solution corresponding to an energy consumption E_{ta} (temperature aware). For the same application we run a task mapping optimization ignoring temperature (temperature unaware task mapping, later referred as TnaTM), which is realized by running the same mapping optimization as described in section IV (Fig. 4), with the difference that a temperature unaware voltage selection is used inside the exploration loop. As discussed in the previous section, this means that energy calculations are based on an "assumed temperature" instead of the actual temperature at which the cores run and, thus, the TnaTM is less efficient. Considering the mapping solution and the voltage levels produced by the TnaTM approach, we run the temperature analysis to obtain the real temperature at which the application will run and, finally, we calculate the consumed energy E_{nta} (not temperature aware). By comparing E_{ta} with E_{nta} we can appreciate the efficiency of using a temperature aware mapping scheme.

Given a certain application, we define the energy efficiency factor G of the TaTM, compared to the TnaTM, as $G = (E_{nta} - E_{ta})/E_{nta} * 100\%$.

Fig. 8 shows the energy efficiency factor obtained for the three application sets. For each set the average and maximum value of G is indicated. It can be observed that, in the case of applications in which task sizes are very similar (set S_1) the energy efficiency factor is smaller. This means that the potential gain of applying a temperature aware approach is larger for the applications with an uneven distribution of task sizes (set S_3).

The explanation for the above phenomenon has its roots in the exponential dependence of leakage current on temperature. As a result, cores running at high temperature will dissipate an unproportionaly high amount of energy. Therefore, a solution in which temperatures are uniformly distributed among cores is, in principle, more energy efficient than one in which some cores run at low and others at high temperature. Balanced temperature distribution is, in principle, related to a balanced distribution of load. Such a balanced distribution is more likely to be obtained with tasks of relatively uniform sizes (set S_1)

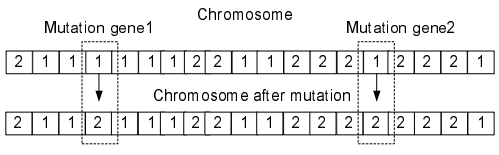

Fig. 7. Multi-point Mutation

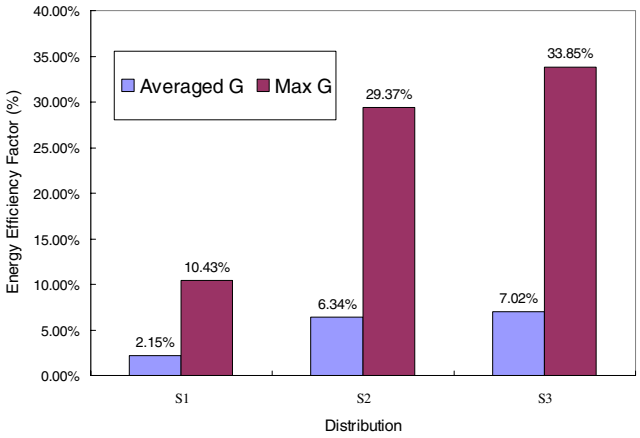

Fig. 8. Task Size Distribution Influence on G

than in the case of tasks that have very different sizes (set S_3). The TaTM approach will actively seek balanced solutions since those will produce lowest energy. The TnaTM approach, however, which ignores the dependence of consumed energy on temperature, can end up with an unbalanced solution that, according to the energy calculation at the "assumed temperature", consumes less energy and also satisfies the time constraints. While in the case of very similar task sizes, it is possible that the solution produced by the TnaTM is relatively balanced, this is much less likely in the case of applications with uneven task size distribution.

In order to confirm our assumption that the main reason for the inefficiency of the TnaTM is the unbalanced temperature distribution, we have drawn the graph in Fig. 9. In order to characterize the temperature variation among cores, for a certain solution, we use the standard deviation. With SD_Tna we denote the standard deviation of core temperatures for a certain solution produced by the TnaTM approach.

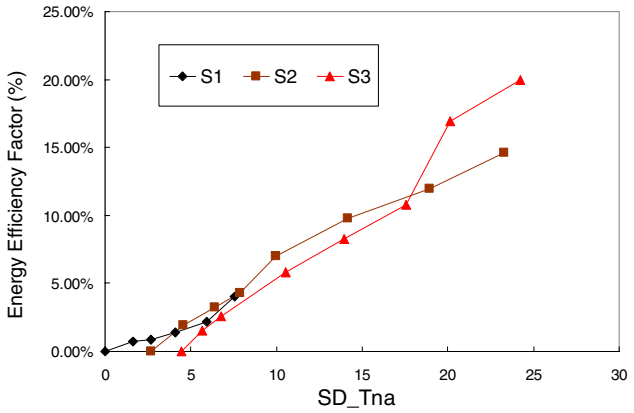

Fig. 9. Relationship Between SD_Tna and G

Fig. 9 shows the average energy efficiency factor G, for the same three sets of applications as in Fig. 8, as a function of SD_Tna. As we can see, the more unbalanced the core temperatures produced by the TnaTM are, the larger the gain is by using TaTM. For some applications in set S_1, the TnaTM produces solutions with balanced (SD_Tna = 0) or close to balanced temperature. For this case the energy efficiency factor is zero or close to zero. For applications in set S_2 and set S_3 the TnaTM will never find balanced temperature solutions.

As mentioned previously, for the TnaTM, it is assumed that the application runs at a certain temperature given by the designer. For our experiments presented so far, we have produced the "assumed temperature" in the following way: for an application we first run the TaTM approach and determine the temperature at which each core is running; we use, as "assumed temperature" for the TnaTM approach, the average of the individual core temperature. This approach is, of course, not applicable in practice, since a designer who does not have a temperature aware design tool will not be able to know the average temperature at which the cores will run. Thus, in reality, the results obtained with a TnaTM approach are worse, compared to the TaTM, than shown in Fig. 8 and Fig. 9. Obviously, E_{nta}, the energy consumption produced by the TnaTM approach, is as larger as further away from reality the designer's "temperature guess" is.

In Fig. 10 we show the average energy efficiency factor for the applications in set S_2, for different "temperature guesses". If the temperature guess of the designer corresponds to the average temperature, as considered in our previous experiments (temperature difference = 0) the average factor G is identical with the one in Fig. 8. If the temperature guess is deviating towards larger or smaller temperatures (which in reality is the case) the discrepancy between the quality of the TaTM and TnaTM approach is increasing.

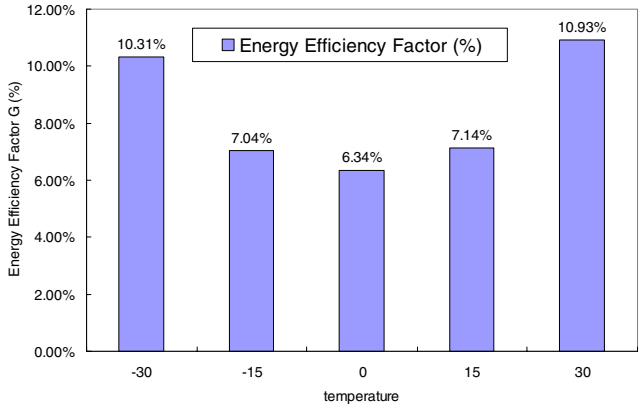

Fig. 10. Temperature Guessing Influence on G

Fig. 11 shows the optimization time needed by our TaTM approach as a function of the application size. As can be observed, even very large applications can be handled in reasonable amounts of time.

We have investigated the efficiency of temperature aware task mapping using two real-life examples: A GSM voice codec and a multimedia MPEG4 audio-video encoder. Details regarding the two applications can be found in [17] and [13], respectively. The GSM voice codec is composed of an encoder

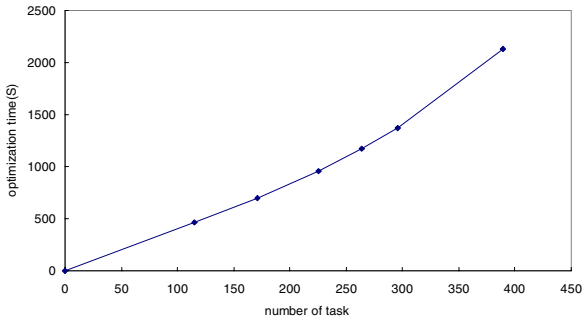

Fig. 11. Optimization Time for TaTM

and a decoder of GSM frames and consists of 87 tasks, and it is considered to be mapped on an architecture composed of 3 cores with 13 voltage modes. The MPEG4 consists of 109 tasks and is also mapped on 3 cores with 13 voltage modes. The results are presented in Fig. 12 and 13, and they confirm the values and trends for the energy efficiency factor G outlined by our previous experiments.

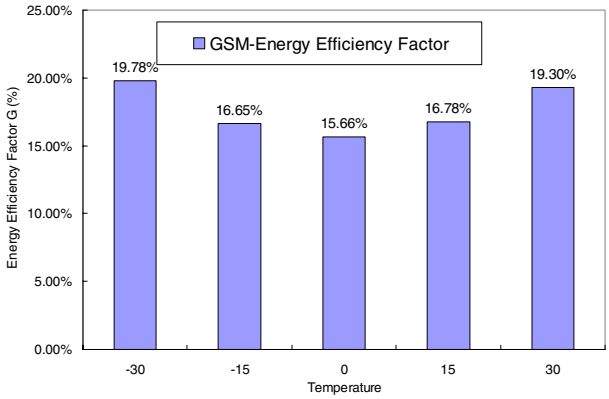

Fig. 12. Real Life Example: GSM

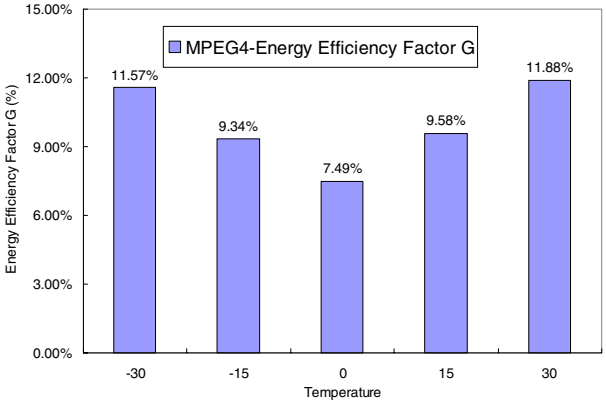

Fig. 13. Real Life Example: MPEG4

VI. Conclusion

In this paper, we propose a temperature aware task mapping approach. We have analysed the potential gain that can be obtained by taking temperature into consideration and discussed main factors which influence the efficiency of the proposed approach. Based on our experiments, we have demonstrated that significant energy gain can be obtained by applying the temperature aware task mapping technique.

References

[1] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and B. M. Al-Hashimi, *Energy Optimization of Multiprocessor Systems on Chip by Voltage Selection*, IEEE Transactions on Very Large Scale Integration Systems, 15(3):262C275, March 2007.

[2] M. Bao, A. Andrei, P. Eles, Z. Peng, *Temperature-Aware Voltage Selection for Energy Optimization*, in Proc. of DATE, Mar. 2008.

[3] W. W. Chu, L. Y. Holloway, M. T. Lan, and K. Efe, *Task Allocation in Distributed Data Processing*, Computer, Vol. 13, No. 11, Nov. 1980, pp. 57-69.

[4] P. Eles, A. Doboli, P. PoP, Z. Peng, *cheduling with bus access optimization for distributed embedded systems*, IEEE Tran. on VLSI, vol. 8, No. 5, pp. 472C491, 2000.

[5] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M. Stan, *HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design*, IEEE onVLSI Systems, 14(5):501-513, 2006.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[7] D.E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison Wesley Publishing Company, 1989.

[8] L-F. Leung, C-Y. Tsui, and W-H. Ki., *Minimizing Energy Consumption of Multiple-Processors-Core Systems with Simultaneous Task Allocation, Scheduling and Voltage Assignment*, the Proc. of Asia South Pacific Design Automation (ASP-DAC), pages 647-652, Jan. 2004.

[9] Y. Li, B. C. Lee, D. Brooks, Z. Hu, K. Skadron, *CMP Design Space Exploration Subject to Physical Constraints*, HPCA06, pp. 15-26, 2006.

[10] W. P. Liao, L. He, and K. M. Lepak, *Temperature and supply voltage aware performance and power modeling at micro-architecture level*, IEEE TonCAD, V24, no. 7, pp. 1042-1053, July 2005.

[11] Y. Liu, H. Yang, R.P. Dick, H. Wang, L. Shang, *Thermal vs Energy Optimization for DVFS-enabled Processors in Embedded Systems*, Int. Symp. on Quality Electronic Design (ISQED07), pp. 204 - 209, 2007.

[12] S. Martin, K. Flautner, T. Mudge, D. Blaauw, *Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads*, ICCAD, pp. 721-725, 2002.

[13] http://ffmpeg.mplayerhq.hu/

[14] M. Ruggiero, P. Gioia, G. Alessio, L. B. M. Milano, D. Bertozzi, and A. Andrei, *Cooperative, Accurate Solving Framework for Optimal Allocation, Scheduling and Frequency Selection on Energy-Efficient MPSoCs*, in SOC, Nov 2006.

[15] K. Sankaranarayanan, S. Velusamy, M.R. Stan, K. Skadron, *A Case for Thermal-Aware Floorplanning at the Microarchitectural Level*, The Journal of Instruction-Level Parallelism, V7, Oct. 2005, pp. 1-16.

[16] M. Schmitz, B. Al-Hashimi, and P. Eles, *Energy- effcient mapping and scheduling for DVS enabled dis- tributed embedded systems*, DATE'02, pp. 514-521.

[17] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*, Kluwer Academic Publisher, 2004.

[18] S. Wang, R Bettati, *Delay Analysis in Temp.-Constrained Hard Real-Time Systems with General Task Arrivals*, RTSS06, pp. 323-334.

[19] Y. Xie and W.-L. Hung, *Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design*, J. VLSI Signal Processing, vol. 45, no. 3, pp. 177C189, Dec. 2006.

[20] Y. Yang, et al., *ISAC: Integrated Space and Time Adaptive Chip-Package Thermal Analysis*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Jan. 2007.

[21] Y. Zhang, X. Hu and D. Chen, *Task scheduling and voltage selection for energy minimization*, Proc. DAC, June 2002.