Energy-Aware Task Allocation onto Unrelated Heterogeneous Multicore Platform for Mixed Criticality Systems

Muhammad Ali Awan, Damien Masson, Eduardo Tovar

Energy-aware Task Allocation onto Unrelated Heterogeneous Multicore Platform for Mixed Criticality Systems

M. Ali Awan*, Damien Masson†, Eduardo Tovar*
*CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
{muaan, emt}@isep.ipp.pt
†LIGM - Laboratoire d'Informatique Gaspard-Monge
damien.masson@univ-paris-est.fr

***Abstract*—Heterogeneous multicore platforms have become an attractive choice to deploy mixed criticality systems demanding diverse computational requirements. One of the major challenges is to efficiently harness the computational power of these multicore platforms while deploying mixed criticality applications. The problem is acerbated with an additional demand of energy efficiency. It is particularly relevant for the battery powered embedded systems. We propose a partitioning algorithm for unrelated heterogeneous multicore platforms to map mixed criticality applications that ensures the timeliness property and reduces the energy consumption.**

I. Introduction

Modern real-time (RT) applications are becoming increasingly complex everyday. These applications may have different criticality levels. A criticality level corresponds to a level of assurance against failure. The highest-criticality functions are vital for the operations of the system and any violation of temporal constraint (fault) may lead to a disastrous consequences. However, fault of lower criticality functions may cause a tolerable decrease in the quality of service.

In the past, applications of different criticality were hosted on separate components. A recent trend in RT and embedded system domain to integrate the functionalities belonging to different criticality levels on a single hardware platform has paved a way towards mixed criticality systems. Mixed criticality systems are common in many domains such as avionics and automotive industry. For example, an unmanned aerial vehicle may have functionalities corresponding to flight critical (concerning the safety of flight) and mission critical (reconnaissance and surveillance) operations. It is vital for the integrity of the aerial vehicle to ensure the safe operation of a flight in all conditions while reconnaissance and surveillance are secondary functionalities.

In order to increase the level of assurance against failure for the high criticality functions, pessimistic estimates of the worst-case-execution time (WCET) are determined using static analysis. However, a mixed criticality system designed with the pessimistic estimates of WCET for the high criticality functions (tasks) will lead to inefficient utilisation of the resources and unnecessary over-provisioning of the system. This is driven by the fact that high criticality WCET estimates are rarely achieved in real execution. To efficiently exploit the available resources Steve Vestal [1] proposed a mixed criticality system model. In this model, tasks have different estimates of WCET corresponding to different criticality levels. A system starts in a low criticality mode and the schedulability of the system is ensured assuming a WCET estimate corresponding to current low criticality mode. If any task exceeds its execution beyond the WCET estimate defined for the given criticality mode of operation, a system transitions to high criticality mode.

Modern multicore platforms provide sufficient computing capabilities to deploy mixed criticality systems on a single chip. Among different kinds of multicore architectures, heterogeneous multicore platforms composed of more than one heterogeneous processing units (cores) has gained popularity to perform specific tasks well and cheap. Various components (applications) in a mixed criticality system have diverse computing requirements that makes heterogeneous multicore platforms an attractive choice to opt. Despite all benefits, efficient mapping of mixed criticality applications on heterogeneous multicore platform is one of the major challenges faced by system designers.

The problem is acerbated with an additional limitation of energy supply. This is particularly relevant in battery powered and nomadic embedded systems. Even when the application is technically feasible upon the targeted platform in the sense that the platform can provide a sufficient computing capacity for the execution of the application, it has become unreasonable to expect to implement such a system without addressing the issue of reducing its energy consumption. The energy consumption of a heterogeneous platform can be reduced in two different ways. Firstly, a task should be mapped to a processor where it dissipates minimum dynamic power. Secondly, the energy-aware scheduling mechanism can be employed to further reduce the energy consumption on each

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER Research Centre); also by FCT/MEC and the EU ARTEMIS JU within project(s) ARTEMIS/0001/2013 - JU grant nr.621429 (EMC2) and ARTEMIS/0003/2012 - JU grant nr.333053 (CONCERTO).

core. This work explores the former technique.

We use partitioned scheduling to map the given set of tasks belonging to different criticality levels on a heterogeneous multicore platform such that available computational capacity of the hardware platform is efficiently exploited, and energy consumption is minimised while ensuring the timeliness properties of the system. This is an NP-hard problem in a strong sense as it is a special case of bin packing and hence, heuristics are the way forward.

There has been some efforts in the state-of-the-art [2]–[5] to propose new techniques and evaluate the potential of existing bin packing heuristic to perform the allocation with an objective to optimise the utilisation of the system. However, the state-of-the-art on the energy minimisation of mixed criticality systems is very limited. Broekaet et al. [6] proposed system level power management approach that allocates energy budgets to virtual machines. In this techniques an overrunning virtual machine deducts its additional budget from budget of low priority virtual machine to be scheduled in future. Legout et al. [7] proposed a technique that provides trade-off between energy reduction and the number of deadline misses of low criticality tasks. Finally, Zhang et al. [8] allocates energy budgets to different cores and perform allocation through genetic algorithm to extend the battery life time of the system. They assume that the energy consumption of a task is only a function of its execution time. We consider a realistic power model in which the energy consumption of a task depends on the characteristics of the core type.

II. System model

This work considers a partitioned unrelated heterogeneous multicore platform $\pi \stackrel{\text{def}}{=} [\pi^1, \pi^2, \ldots, \pi^M]$ composed of M distant cores. We adopt a common mixed criticality task model, initially proposed by Steve Vestal [1] and later extended by other researchers in the RT research community [9]. We consider a dual criticality system in which a system executes in either low criticality mode (L-mode) or high criticality mode (H-mode) of operation. We assume a set of n independent sporadic tasks $\tau \stackrel{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task $\tau_i \stackrel{\text{def}}{=} \langle T_i, D_i, \kappa_i, \vec{C_i}(H), \vec{C_i}(L), \vec{E_i} \rangle$ is characterised by its minimum inter-arrival time T_i, deadline D_i, criticality level κ_i, a vector of WCET profile $\vec{C_i}(H)$ in H-mode of operation, a vector of WCET profile $\vec{C_i}(L)$ in L-mode of operation and a vector of energy consumption profile $\vec{E_i}$. Each task τ_i belongs to either high criticality level (H-task) or low criticality level (L-task), i.e., $\kappa_i \in \{L, H\}$. The WCET profiles $\vec{C_i}(H) \stackrel{\text{def}}{=} (C_i^1(H), C_i^2(H), \ldots, C_i^M(H))$ and $\vec{C_i}(L) \stackrel{\text{def}}{=} (C_i^1(L), C_m^2(L), \ldots, C_i^M(L))$ are the WCET estimates in the H-mode and L-mode of operation, respectively, on different cores indexed from 1 to M. Similarly, energy consumption profile $\vec{E_i} \stackrel{\text{def}}{=} (E_i^1, E_m^2, \ldots, E_i^M)$ represents the energy consumption of task τ_i on different cores in L-mode.

It is very common to assume in state-of-the-art that energy consumption of a task is a function of its execution time. However, in reality, energy consumption on a certain processor depends also on the set of instructions it has to execute to perform the desired functionality. Various instructions use different parts of CPU, and hence may result in a different estimate of energy consumption. Therefore, two applications with identical execution time may consume different energy depending on the characteristics of the instructions used, and the number of cache misses involved. In this work, we consider a realistic power model in which the energy consumption of a task depends on the characteristics of the core type and hence, computed on each core using existing energy measurement technique [10]. This approach has the flexibility to incorporate the effect of other system resource usage on energy consumption such as memory and caches etc.

We assume that $\tau(L) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in L\}$ and $\tau(H) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in H\}$ represent the subsets of low- and high-criticality tasks in τ, respectively. It is assumed that $\forall \tau_i \in \tau(H) \wedge \forall m \in [1, 2, \ldots, M]$, $C_i^m(L) \leq C_i^m(H)$ and $\forall \tau_i \in \tau(L) \wedge \forall m \in [1, 2, \ldots, M]$, $C_i^m(H) = 0$. Tasks are not allowed to migrate after assignment as we assume partitioned scheduling. The schedulability analysis proposed by Ekberg and Yi [11] is used to test the feasibility of the system in both low and high criticality mode. This analysis shortens the deadlines of $\tau(H)$ in the L-mode to keep the schedule ahead of time which allows a safe transition from L to H-mode. The utilisation of task τ_i in low and high criticality mode on processor π^m are represented as $U_i^m(L) \stackrel{\text{def}}{=} \frac{C_i^m(L)}{T_i}$ and $U_i^m(H) \stackrel{\text{def}}{=} \frac{C_i^m(H)}{T_i}$, respectively. A subset of task assigned to a core π^m is represented as $\tau(\pi^m)$. The overall utilisation of tasks assigned to core π^m in L and H-mode is defined as $U^m(L) \stackrel{\text{def}}{=} \sum_{\forall \tau_i \in \tau(\pi^m)} C_i^m(L)$ and $U^m(H) \stackrel{\text{def}}{=} \sum_{\forall \tau_i \in \tau(\pi^m)} C_i^m(H)$, respectively. We assume an arbitrary deadline model in our system in which task deadline has not relation with its minimum inter-arrival time.

III. Preliminaries

Initially, we define the following concepts needed to explain the proposed allocation heuristics.

Definition 1 (Energy density ED_i^m): The energy density of a task τ_i on a core π^m is defined as $ED_i^m \stackrel{\text{def}}{=} \frac{E_i^m}{T_i}$, where E_i^m corresponds to the energy consumption of task in the low criticality mode. This is a measure of the average power dissipation in the low criticality mode.

Definition 2 (Energy density difference EDD_i^m): The energy density difference of a task τ_i on a core π^m is defined as $EDD_i^m \stackrel{\text{def}}{=} \min\{ED_i^h : h \neq m \wedge ED_i^h \geq ED_i^m\} - ED_i^m$. This value corresponds to the difference between the next higher energy density of τ_i on any other core and the energy density on the current core.

Definition 3 (Low criticality utilisation difference LUD_i^m): The low criticality utilisation difference of a task τ_i on a core π^m is defined as $LUD_i^m \stackrel{\text{def}}{=} \min\{U_i^h(L) : h \neq m \wedge U_i^h(L) \geq$

$U_i^m(L)\} - U_i^m(L)$. This value is the difference between the next higher low criticality utilisation of τ_i on any other core and the low criticality utilisation on the current core.

Definition 4 (High criticality utilisation difference HUD_i^m): The high criticality utilisation difference of a task $\tau_i \in \tau(H)$ on a core π^m is defined as $HUD_i^m \stackrel{\text{def}}{=} \min\{U_i^h(H) : h \neq m \wedge U_i^h(H) \geq U_i^m(H)\} - U_i^m(H)$. This measure represents the difference of next higher high criticality utilisation of $\tau_i \in \tau(H)$ on any other core and the high criticality utilisation on the current core.

Initially, we compute ED_i^m and LUD_i^m for all tasks, and HUD_i^m for high criticality tasks (i.e., $\forall \tau_i \in \tau(H)$) on each core. Afterwards, we generate the following three lists out of these values.

1) Assume a set LUD such that $LUD \stackrel{\text{def}}{=} \left\{LUD_1^{min}, LUD_2^{min}, LUD_3^{min}, \ldots, LUD_{|\tau|}^{min}\right\}$, where $LUD_i^{min} \stackrel{\text{def}}{=} \min_{\forall m}\{LUD_i^m\}$. S^{LUD} is a list that contains a sequence of numbers of LUD sorted in a non-increasing order.
2) Assume a set $HT \stackrel{\text{def}}{=} \left\{HUD_{j_1}^{min}, HUD_{j_2}^{min}, \ldots, HUD_{j_{|\tau(H)|}}^{min}\right\}$, where $HUD_{j_x}^{min} \stackrel{\text{def}}{=} \min_{\forall m}\{HUD_{j_x}^m\}$ and $\tau_{j_x} \in \tau(H)$. Similarly, assume another set $LT \stackrel{\text{def}}{=} \left\{LUD_{k_1}^{min}, LUD_{k_2}^{min}, \ldots, LUD_{k_{|\tau(H)|}}^{min}\right\}$, where $\tau_{k_x} \in \tau(L)$. Let S^{HT} and S^{LT} represent the sequence of elements of set HT and LT, respectively, sorted in non-increasing order, then, S^{HUD} is a list that concatenates S^{HT} and S^{LT}, i.e., $S^{HUD} \stackrel{\text{def}}{=} S^{HT}||S^{LT}$, where $||$ represents the concatenation sign.
3) Assume that EDD is a set defined as $EDD \stackrel{\text{def}}{=} \left\{EDD_1^{min}, EDD_2^{min}, EDD_3^{min}, \ldots, EDD_{|\tau|}^{min}\right\}$, where $EDD_i^{min} \stackrel{\text{def}}{=} \min_{\forall m}\{EDD_i^m\}$. S^{EDD} is a list that contains all the elements of EDD sorted in non-increasing order.

IV. Proposed approach

In our proposed approach, we compute a set of feasible allocations and select the one that minimises the overall energy consumption of the system. Each feasible allocation ensures the schedulability of assigned tasks on each core both in LOW and HIGH criticality modes. The following method is used to collect the feasible allocations. A suffrage based task-to-core allocation scheme ILLED (proposed by Awan et al. [**?**] for single criticality systems) ranks the tasks based on a metric called density difference[1] while performing allocation to their preferred core. The density difference of a task shows how much a system will lose in term of given criteria (utilisation or energy) if a task is not allocated to its preferred core. The ILLED algorithm ranks tasks based on density difference metric and performs the allocation to optimise the given criteria (utilisation or energy). This ranking plays an important role in the allocation process. The tasks ranked higher have the higher probability of getting mapped to their preferred core. The proposed allocation heuristics manipulates S^{LUD}, S^{HUD} and S^{EDD} density difference lists to obtain a set of feasible task-to-core mappings. The ILLED algorithm signals a failure if the mapping is not schedulable[2] in either high criticality or low criticality mode.

The pseudocode of the proposed approach is given in Algorithm 1. First of all, we perform the task-to-core mapping based on a list S^{EDD} through ILLED algorithm. If the task-to-core mapping is successful, meaning it ensures the schedulability in both high criticality and low criticality modes, we are done. As the ordering of the tasks in S^{EDD} is based on the energy density difference, therefore, ILLED generates a mapping to minimise the energy consumption. However, if a system is not schedulable in either of high or low criticality mode, we need to determine an order of a list that leads to a feasible task-to-core mapping ensuring schedulability in both modes.

In mixed criticality systems, it is very important to ensure the schedulability of the high criticality tasks in both low and high criticality modes. A conservative allocation approach gives preference to the high criticality tasks in the assignment process to ensures their schedulability in both modes. Such an allocation can be achieved through S^{HUD} list. Assume, we divide the list S^{HUD} into two parts S^H and S^L corresponding to high and low criticality tasks respectively, and apply the ILLED algorithm on S^H followed by S^L.

It is also desirable to ensure the schedulability of the low criticality tasks in a low criticality mode. In practice, mixed criticality system mostly stays in low criticality mode and occasionally transitions into a high criticality mode triggered due to an overrun or other external/internal events. Therefore, it is more beneficial to optimise the energy efficiency in the low criticality mode. An allocation based on a list S^{LUD} ordering maximises the schedulability in the low criticality mode as it is sorted with respect to low criticality utilisation. However, there may be a possibility that system may not be feasible in high criticality mode when an allocation is performed with such an ordering (S^{LUD}). Therefore, we start with a list S^{LUD} in our allocation process and gradually promote the high criticality tasks in our S^{LUD} list corresponding to the ordering given in S^{HUD} to generate all feasible allocations.

The reordering mechanism is explained as follows. We perform the allocation with respect to the given list and compute its energy consumption if the allocation is successful (schedulable both in low and high mode). Afterwards, we modify S^{LUD} list, by promoting the entry of high criticality task by one position to get a new list. As modification in S^{LUD} list is performed with respect to the list S^{HUD}, therefore, we select the high criticality task corresponding to the entry on

[1]The density difference is a generic term used to indicate any of the following quantities, i) low criticality utilisation difference, ii) high criticality utilsation difference or iii) energy density difference.

[2]The ILLED algorithm is initially designed for single criticality system. To make it compatible with mixed criticality systems, we replaced the feasibility test of the ILLED algorithm with Ekberg and Yi's schedulability analysis [11].

Algorithm 1 Pseudocode of the proposed algorithm

Input: τ, π
Output: Assignment
1: Compute S^{EDD}, S^{LUD} and S^{HUD}
2: Assume A[i] represent the i^{th} assignment
3: A[0] := Perform the ILLED allocation with S^{EDD}
4: **if** (A[0] is feasible) **then return** A[0]
5: **end if**
6: $i := 0$
7: **while** (true) **do**
8: A[i] := Perform the ILLED allocation with S^{LUD}
9: **if** (A[i] is feasible) **then**
10: E[i] := Compute energy consumption of A[i]
11: $i := i + 1$
12: **end if**
13: Promote high criticality task's entry in S^{LUD}
14: **if** (Promotion fails) **then**
15: break
16: **end if**
17: **end while**
18: A[i] := Perform the ILLED allocation with S^{HT} followed by S^{LT},
19: **if** (A[i] is feasible) **then**
20: $E[i]$:= Compute energy consumption of A[i]
21: **end if**
22: index := Find index that gives $\min_{\forall i}(E[i])$
23: **return** A[index]

top of the S^{HUD} list and promote its corresponding entry in S^{LUD} by one position. If the task corresponding to the top most element in S^{HUD} is also at the same level as in S^{LUD}, we need to select the task corresponding to the second element of S^{HUD} in S^{LUD} and promote it by one position in S^{LUD} unless it reaches the same position as in S^{HUD}. As mentioned previously, after every modification we have to compute the energy consumption, if the allocation is successful. After all these promotions of high criticality tasks, S^{LUD} will have the same ordering as given in S^{HUD}. However, S^{LUD} does not represent the same list as S^{HUD} as high criticality tasks are ordered with respect to HUD_i^m in S^{HUD}. Therefore, as a final step, we perform the allocation based on a list S^{HUD} and compute the energy consumption on successful allocation. The allocation with a list S^{HUD} is performed in a slightly different manner. We first divide the list S^{HUD} into two lists S^{HT} and S^{LT} representing high and low criticality tasks. All elements in S^{HT} and S^{LT} are sorted in non-increasing order with respect to HUD_i^m and LUD_i^m respectively. First, we perform the allocation of S^{HT} and then on top of this we perform the allocation of low criticality task from a list S^{LT}. The energy consumption is computed on successful allocation and stored in a variable E^{HUD}. Let E^{LUD} be the minimum of the energy consumptions computed for all the iteration performed on a list S^{LUD}. A minimum of E^{HUD} and E^{LUD} will give us our desired allocation that ensures the schedulability and reduces the total energy consumption.

Finally, we explain an adapted ILLED algorithm [**?**]. This algorithm assigns the tasks to their favourite core in the specified order of the provided list. If a task cannot be assigned to its preferred core, the current entry of the task in the provided list is updated with the value on next preferred core in the order. The entry corresponding to this task is added again in the provided list on its appropriate location based on its new updated value. On a successful assignment the task from the provided list is removed. This allocation scheme should continue unless the provided list is empty or a task cannot be assigned to any core. We used Ekberg and Yi's [11] test to check the schedulability of a core after mapping any task.

V. Conclusion and future directions

This work presents the energy-aware task-to-core allocation for mixed criticality systems deployed on heterogeneous multicore platform considering a realistic power model. Initial assessment of this algorithm predicted upto 8.9% gains over the traditional first-fit bin packing heuristics in terms of energy consumption and scheduled task-set. In the future, it is intended to evaluate the performance of the proposed approach for different types of applications and hardware platforms. Another possible extension is to integrate the effect of leakage power dissipation in the proposed approach.

References

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International, Dec 2007, pp. 239–243.

[2] K. Lakshmanan, D. de Niz, R. Rajkumar, and G. Moreno, "Resource allocation in distributed mixed-criticality cyber-physical systems," in Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, June 2010, pp. 169–178.

[3] D. TamasSelicean and P. Pop, "Design optimization of mixed-criticality real-time applications on cost-constrained partitioned architectures," in Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd, Nov 2011, pp. 24–33.

[4] O. Kelly, H. Aydin, and B. Zhao, "On partitioned scheduling of fixed-priority mixed-criticality task sets," in Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, Nov 2011, pp. 1051–1059.

[5] P. Rodriguez, L. George, Y. Abdeddaïm, and J. Goossens, "Multi-criteria evaluation of partitioned edf-vd for mixed-criticality systems upon identical processors," in Workshop on Mixed Criticality Systems, 2013.

[6] L. s. Florian Broekaert, Agnes Fritsch and S. Tverdyshev, "Towards power-efficient mixed critical systems," OSPERT, vol. 2013, pp. 30–35, 2013.

[7] V. Legout, M. Jan, and L. Pautet, "Mixed-Criticality Multiprocessor Real-Time Systems: Energy Consumption vs Deadline Misses," in First Workshop on Real-Time Mixed Criticality Systems (ReTiMiCS), Taipei, Taiwan, Aug. 2013, pp. 1–6.

[8] X. Zhang, J. Zhan, W. Jiang, Y. Ma, and K. Jiang, "Design optimization of security-sensitive mixed-criticality real-time embedded systems," in 1st workshop on Real-Time Mixed Criticality Systems (ReTiMiCS), 2013.

[9] A. Burns and R. Davis, "Mixed criticality systems-a review," Department of Computer Science, University of York, Tech. Rep, 2013.

[10] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: A platform for os-level power management," in Proceedings of the 4th ACM European Conference on Computer Systems, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 289–302.

[11] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," Real-Time Systems, vol. 50, no. 1, pp. 48–86, 2014.