Performance and Energy Trade-Offs in Heavy-Light Platforms

Vasileios Glykantzis, Andres Gomez, Rehan Ahmed, Lothar Thiele
glykantv@student.ethz.ch, {gomeza,rahmed,thiele}@tik.ee.ethz.ch
ETH Zurich

Abstract

While researchers have been studying ways to minimize the energy consumed in embedded platforms for many years, the costs of implementing advanced energy minimization techniques (e.g. fine-grained DVFS) are now prohibitive in low-cost microcontroller platforms. In this work, we study the energy-performance tradeoffs in the recently proposed Heavy-Light architecture, which consists of a power-hungry (Heavy) core and a low-power (Light) core. The former is designed with performance guarantees under worst-case PVT conditions, while the latter is not. We develop and formally prove the optimality of task allocation policies with respect to energy and makespan. We show that for low levels of parallelism, there is a trade-off between minimizing makespan and energy consumption (i.e. minimizing one does not minimize the other). Theoretical results were experimentally validated by emulating the heavy-light behavior in a commercially available platform. The results show up to 50% decrease in makespan and up to 22% energy savings compared to a single-core platform, depending on the workload and platform characteristics.

I. Introduction

Over the past decade, there have been a large research effort to reduce the energy consumption in embedded systems. Heterogeneous architectures have emerged as a viable candidate in high performance embedded systems. One example is the heterogeneous big.LITTLE architecture [1], which consists of a big cluster, for performance, and a LITTLE cluster, for energy efficiency. Even though the two clusters are microarchitecturally different, they support the same instruction set architecture (ISA), simplifying their programming. In order to fully exploit their energy saving potential, these architectures require an advanced HW/SW infrastructure, such as fine-grained Dynamic Voltage and Frequency Scaling (DVFS), memory virtualization, multi-threading, thread migration and a full-featured operating system. All of these are beyond the reach of low-end MCUs.

On the other end of the spectrum, the low-end microcontroller market is currently dominated by simple, cache-less, single-core platforms optimized for low power and cost. Those advanced energy saving techniques from high-end systems are not supported. Consequently, these systems typically rely on frequency scaling and shutdown for minimizing their energy consumption. More recently, two important trends in the microcontroller domain have opened new avenues for tackling the energy consumption problem. First, heterogeneous dual-core architectures have been shown to reduce energy when compared to single-core platforms [2]. These systems, with different microarchitectures and ISAs, can exhibit high energy efficiency with certain applications. However, one limitation is that due to different ISAs, tasks are (statically) allocated at design time, limiting dynamic (online) decisions.

To circumvent this limitation, the semiconductor industry has started focusing on dual-core platforms with different model guardbands for different cores. Reduced guardbands lead to reduced core area and increased energy efficiency. Recent studies have shown that a dual-core platform consisting of one (Heavy) core designed to work reliably under the worst-case conditions and another (Light) core designed using reduced design margins, can reduce the platform's energy consumption by up to 20% compared to a single-core designed for worst-case conditions [3]. Due to uniform ISA, task allocation in these platforms does not have to be static/fixed.

In this work, we study the tradeoff between minimizing energy and maximizing performance when executing an application on a Heavy-Light platform. Heavy-Light platforms are restrictive in terms of the energy saving mechanisms available:no DVFS or fine-grained power gating can be applied. As is typical in microcontroller-based platforms, there is also no multi-threading or thread migration due to their excessive cost. Therefore, our proposed solutions are task allocation policies geared towards minimizing energy or maximizing performance. We prove that for Heavy-Light platforms, there is an assignment of tasks to cores, which minimizes both energy consumption and application makespan. Therefore, for the optimal case, there is no tradeoff between energy and performance. Furthermore, we prove that even when the optimal task assignment/partitioning cannot be achieved, there is no tradeoff between energy and performance when task level parallelism is high. We evaluate the proposed concepts by detailed simulations and evaluation on a hardware testbed.

II. Related Works

Optimizing an embedded system's energy and performance have been goals pursued by designers for decades. Generally speaking, there are two types of knobs designers can adjust to minimize energy: 1) Dynamic Voltage and Frequency Scaling, and 2) Dynamic Power Management. In the first, a core's voltage and frequency can be reduced to lower its power consumption and speed. In the second, cores are given different states, i.e. active and idle, which reduce the core's average power consumption. Task allocation and scheduling for minimizing energy using these knobs can be formulated as optimization problems. Unfortunately, most of these problems are known to be NP-complete/hard. Consequently, many heuristics, some with optimality bounds, have been proposed to estimate optimal solutions.

In [4–6], the authors propose optimal energy minimization algorithms along with bounded heuristics under real-time task constraints. The proposed solutions use speed scaling and are restricted to uni-core platforms. In heterogeneous MPSoCs, complex scheduling and load balancing mechanisms have been proposed [7]. For big.LITTLE architectures, other works [8] have studied utilization-aware load balancing techniques and reported energy savings of 11.35% compared to the standard Linux schedulers.

In [9], the authors propose workload allocation heuristics to minimize the energy of a real time application. The authors use a combination of optimization formulations and heuristics to find efficient solutions. In [10], the authors propose a race-to-idle scheme, where the system is placed in *sprint*/high performance mode when it has tasks to execute. The system is placed in *sleep* mode when the tasks finish execution. The authors show that the proposed scheme can give significant energy and performance improvements under certain platform configuration and application scenarios. In [11], a hardware platform for evaluating DVFS at the microcontroller level is presented. While it is shown that a MCU reduces its energy at lower voltages, this is achieved by using costly and inefficient external LDO regulators.

When executing a set of parallelizable instructions in a given microarchitecture, voltage and frequency, multi-core platforms always consume more power than a single-core. We will show that our energy minimization schemes, the single-microarchitecture Heavy-Light platform can achieve significant energy savings with respect to a single-core without commonly used techniques such as fine-grained DVFS. First we present a simplified task model and partitioning scheme which reduces energy and makespan minimization to a mapping problem. Depending on the task-set's available parallelism, our proposed allocation policy is proven to be optimal for both objectives. Our minimization accounts not just for the energy consumed by the cores, but also by on-chip peripherals, which can be a substantial portion of the power budget in low-cost microcontroller systems.

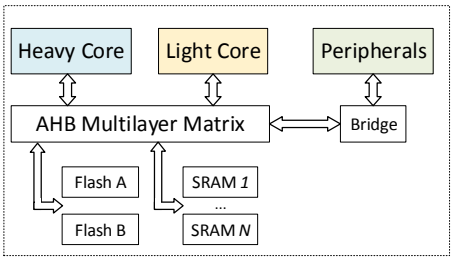


Fig. 1. Architecture of proposed dual-core platform.

III. System and Task Model

Processor Model: The platform considered consists of two cores, as shown in Figure 1. Both cores have same microarchitecture, but different power consumption due to different timing closures. The *Heavy Core* (HC) is designed under worst-case design margins and is guaranteed to work at the maximum frequency F_{HC}. The *Light Core* (LC) is designed with reduced design margins and is not guaranteed to operate at F_{HC}. However, it is guaranteed to operate at a lower frequency $F_{\mathrm{LC}} \leq F_{\mathrm{HC}}$.

Power Model: The system power consumption can be divided into three main components: the power consumption of the HC (P_{HC}), the power consumption of the LC (P_{LC}) and the power consumption of the system peripherals (P_{Sys}). Due to reduced design margins, LC is more energy efficient compared to the HC. This implies that even when HC and LC operate at the same frequency, $P_{\mathrm{LC}} \leq P_{\mathrm{HC}}$. Each component has two different states: *active* and *sleep*. The sleep power of all components is non-negligible; however, it is constant. This constant factor is ignored in all power/energy computations in the remaining paper. Therefore, $P_{\mathrm{X}} = P_{\mathrm{X,Active}} - P_{\mathrm{X,Sleep}}$, $X \in \{\mathrm{HC}, \mathrm{LC}, \mathrm{SYS}\}$. The peripherals can be put into sleep only when both cores are in sleep state. Formally, we focus on Heavy-Light platforms that operate under the following three conditions:

Condition 1: $F_{\mathrm{HC}} \geq F_{\mathrm{LC}} \geq \dfrac{F_{\mathrm{HC}}}{2}$

Condition 2: $\dfrac{F_{\mathrm{LC}}}{P_{\mathrm{LC}}} \geq \dfrac{F_{\mathrm{HC}}}{P_{\mathrm{HC}}}$

Condition 3: $P_{\mathrm{HC}} \leq P_{\mathrm{LC}} + P_{\mathrm{Sys}}$

Condition 1 emphasizes that LC might have performance penalty, due to its relaxed design margins. Condition 2 implies that LC is more energy efficient compared to HC, given that sleep powers are much smaller compared to active powers. Finally, condition 3 states that the power consumption of the peripherals is comparable to the power consumption of the HC. Previous works have shown that low power MCU platforms satisfy these conditions [2, 3].

Task Model: Applications consist of a set of n periodic

and independent tasks. It is assumed that all tasks have the same period D. Furthermore, it is assumed that all tasks are atomic units of execution; thus, preemptions and migrations are not allowed. This is a fairly restrictive assumption; however, it is realistic for deeply low power MCU platforms. C_i is used to denote the computation cycles of task i.

IV. Theoretical Results

Since all tasks are atomic and have same deadline/period, the scheduling problem reduces to partitioning tasks to HC and LC with the objective of minimizing makespan or minimizing energy consumption. As we will see later in this section, for certain task and system configurations, these two objectives are conflicting. However, in most scenarios, these two objectives lead to the same solution i.e. minimizing makespan also minimizes energy. Formally, we can state the following optimization formulation:

Variables:

$$\alpha_{i,X} = \begin{cases} 1, & \text{if task } i \text{ is assigned to core } X \in \{\text{HC}, \text{LC}\} \\ 0, & \text{otherwise} \end{cases}$$

Objectives:

$$\text{Obj 1: Minimize}\big\{\max\{A_{\text{HC}}, A_{\text{LC}}\}\big\} \quad (1)$$

$$\text{Obj 2: Minimize}\big\{P_{\text{HC}}A_{\text{HC}} + P_{\text{LC}}A_{\text{LC}} + P_{\text{Sys}}A_{\text{sys}}\big\} \quad (2)$$

$$\text{where: } A_{\text{X}} = \sum_{1 \le i \le n} \frac{\alpha_{i,\text{X}} \cdot C_i}{F_X}, X \in \{\text{HC}, \text{LC}\}$$

$$A_{\text{Sys}} = \max\{A_{\text{HC}}, A_{\text{LC}}\}$$

Constraints:

$$\sum_{X \in \{\text{HC},\text{LC}\}} \alpha_{i,X} = 1 \qquad \forall 1 \le i \le n \quad (3)$$

$$\max\{A_{\text{HC}}, A_{\text{LC}}\} \le D \quad (4)$$

Fig. 2. Mixed Integer Linear Programming (MILP) formulation for the partitioning phase.

Objective 1 is minimizing makespan of the application while objective 2 is minimizing the energy consumption during one period D. The constraints specify that each task must be assigned to the HC or LC and that the application's makespan must be less than D.

In the following section, we present theoretical results assuming that task-to-core partitioning is fixed/given. Later we will combine these theoretical results with a partitioning heuristic to propose our energy/performance oriented schemes. Assuming fixed partitioning, we get two *task bins* of possibly different sizes. We denote the total computation cycles of all tasks in the large bin and small bin by C_{Lg} and C_{Sm} respectively ($C_{\text{Lg}} \ge C_{\text{Sm}}$). We also use an auxiliary variable $\Delta = \frac{C_{\text{Lg}} - C_{\text{Sm}}}{C_{\text{Lg}} + C_{\text{Sm}}}$ to quantify the imbalance between bins. $\Delta = 0$ when both bins are equal and it approaches 1 as the imbalance between bins increases. Now we study the characteristics of task partitioning which minimizes makespan/energy.

Theorem 1 *The makespan and the energy consumption are simultaneously minimized when $\Delta = \Delta_{\text{opt}}$, where:*

$$\Delta_{\text{opt}} = \frac{F_{\text{HC}} - F_{\text{LC}}}{F_{\text{HC}} + F_{\text{LC}}} \quad (5)$$

and the large/small task bins are executed in HC/LC respectively.

Proof: When $\Delta = \Delta_{\text{opt}}$, $\frac{C_{\text{HC}} - C_{\text{LC}}}{C_{\text{HC}} + C_{\text{LC}}} = \frac{F_{\text{HC}} - F_{\text{LC}}}{F_{\text{HC}} + F_{\text{LC}}}$. This can be algebraically reduced to $\frac{C_{\text{Lg}}}{F_{\text{Lg}}} = \frac{C_{\text{Sm}}}{F_{\text{Sm}}}$. L.H.S is A_{HC} when HC executes the large task bin and R.H.S is A_{LC} when LC executes the small task bin. Therefore, the active times of both cores are equal. Increasing the computation cycles of LC or HC, by shifting tasks, would increase the corresponding active time; thus increasing makespan. Therefore $\Delta = \Delta_{\text{opt}}$ leads to the optimal makespan. Now, we prove that $\Delta = \Delta_{\text{opt}}$ with HC/LC executing large/small task bins also minimizes energy. Let us suppose that A^* represents the optimal value of makespan. For this value, the energy consumption of the system during one period D is:

$$E^* = A^* \cdot (P_{\text{HC}} + P_{\text{LC}} + P_{\text{Sys}})$$

Now suppose that from this optimal scenario, $0 < x \le A^* \cdot F_{\text{LC}}$ computation cycles are shifted from the LC to HC. The new energy consumption is:

$$E_1 = (A^* + \frac{x}{F_{\text{HC}}})(P_{\text{HC}} + P_{\text{Sys}}) + (A^* - \frac{x}{F_{\text{LC}}}) \cdot P_{\text{LC}}$$

Substracting E^* from E_1, we get $x/F_{\text{HC}} \cdot (P_{\text{HC}} + P_{\text{Sys}}) - x/F_{\text{LC}} \cdot P_{\text{LC}}$; which is always positive due to condition 2. Therefore, $E_1 \ge E^*$.

Conversely, now suppose that from this optimal scenario, $0 < x \le A^* \cdot F_{\text{HC}}$ computation cycles are shifted from the HC to LC. The new energy consumption is:

$$E_2 = (A^* - \frac{x}{F_{\text{HC}}}) \cdot P_{\text{HC}} + (A^* + \frac{x}{F_{\text{LC}}})(P_{\text{LC}} + P_{\text{Sys}})$$

Subtracting E^* from E_2, we get $x/F_{\text{LC}} \cdot (P_{\text{LC}} + P_{\text{Sys}}) - x/F_{\text{HC}} \cdot P_{\text{LC}}$; which is always positive due to conditions 1 and 3. Therefore, $E_2 > E^*$. Consequently, E^* is the minimum possible energy consumption for a given taskset. ■

Given fixed partitions, we can execute tasks in the following ways:

1. *Scenario 1*: Large task bin executed on HC and small task bin executed on LC.
2. *Scenario 2*: Small task bin executed on HC and large task bin executed on LC.
3. *Scenario 3*: Both task bins executed on HC.
4. *Scenario 4*: Both task bins executed on LC.

We now study the conditions under which these execution scenarios are performance/energy optimal.

Theorem 2 *Executing large task bin on HC and small task bin on LC (Scenario 1) is always performance optimal.*

Proof: It is trivial to prove that scenario 1 has lower makespan compared to scenario 2 and scenario 3 has lower makespan compared to scenario 4. We will now prove that Scenario 1 outperforms Scenario 3. i.e. scenario 1 makespan $\leq$ scenario 3 makespan:

$$\max\{C_{\text{Lg}}/F_{\text{HC}},\ C_{\text{Sm}}/F_{\text{LC}}\} \leq (C_{\text{Lg}} + C_{\text{sm}})/F_{\text{HC}}$$

The first term in the max is always less than RHS. Regarding the second term in the max, $C_{\text{Sm}}/F_{\text{LC}} \leq$ RHS given condition 1. ■

We will now derive conditions for an *energy optimal* scheme. The total energies for the four scenarios are calculated using following equations:

$$\begin{aligned}
E_{\text{Total},1} =& P_{\text{HC}} \cdot C_{\text{Lg}}/F_{\text{HC}} + P_{\text{LC}} \cdot C_{\text{Sm}}/F_{\text{LC}} \\
&+ P_{\text{Sys}} \cdot \max(C_{\text{Lg}}/F_{\text{HC}},\ C_{\text{Sm}}/F_{\text{LC}}) \\
E_{\text{Total},2} =& P_{\text{HC}} \cdot C_{\text{Sm}}/F_{\text{HC}} + (P_{\text{LC}} + P_{\text{Sys}}) \cdot C_{\text{Lg}}/F_{\text{LC}} \\
E_{\text{Total},3} =& (P_{\text{HC}} + P_{\text{Sys}}) \cdot (C_{\text{Lg}} + C_{\text{Sm}})/F_{\text{HC}} \\
E_{\text{Total},4} =& (P_{\text{LC}} + P_{\text{Sys}}) \cdot (C_{\text{Lg}} + C_{\text{Sm}})/F_{\text{LC}}
\end{aligned}$$

Theorem 3 *It is always energy optimal to parallelize workload.*

Proof: We first prove that $E_{\text{Total},1} \leq E_{\text{Total},3}$. We then prove that $E_{\text{Total},2} \leq E_{\text{Total},4}$, proving that parallelizing workload always leads to lower energy consumption.

$$\begin{aligned}
E_{\text{Total},1} - E_{\text{Total},3} =& P_{\text{LC}} \cdot \frac{C_{\text{Sm}}}{F_{\text{LC}}} + P_{\text{Sys}} \cdot \max(\frac{C_{\text{Lg}}}{F_{\text{HC}}},\ \frac{C_{\text{Sm}}}{F_{\text{LC}}}) \\
&- P_{\text{HC}} \cdot \frac{C_{\text{Sm}}}{F_{\text{HC}}} - P_{\text{Sys}} \cdot \frac{C_{\text{Lg}} + C_{\text{Sm}}}{F_{\text{HC}}}
\end{aligned}$$

By Theorem 2, $\max\{C_{\text{Lg}}/F_{\text{HC}},\ C_{\text{Sm}}/F_{\text{LC}}\} \leq \frac{C_{\text{Lg}}+C_{\text{sm}}}{F_{\text{HC}}}$. Therefore:

$$E_{\text{Total},1} - E_{\text{Total},3} \leq P_{\text{LC}} \cdot \frac{C_{\text{Sm}}}{F_{\text{LC}}} - P_{\text{HC}} \cdot \frac{C_{\text{Sm}}}{F_{\text{HC}}}$$

R.H.S is always $\leq$ 0 due to condition 2, therefore $E_{\text{Total},1} \leq E_{\text{Total},3}$. We now prove $E_{\text{Total},2} \leq E_{\text{Total},4}$:

$$E_{\text{Total},2} - E_{\text{Total},4} = P_{\text{HC}} \cdot \frac{C_{\text{Sm}}}{F_{\text{HC}}} - (P_{\text{LC}} + P_{\text{Sys}}) \cdot \frac{C_{\text{Sm}}}{F_{\text{LC}}}$$

R.H.S. $\leq$ 0 if conditions 1 and 3 are valid. ■

We now derive conditions where the scenario 1 and scenario 2 are energy optimal.

Theorem 4 $E_{\text{Total},1} \leq E_{\text{Total},2}$ ***iff:*** $\Delta \leq \max\{\Delta_{\text{opt}},\ \Delta_{\text{diff}}\}$, *where:*

$$\Delta_{\text{diff}} = \frac{P_{\text{Sys}}(F_{\text{HC}} - F_{\text{LC}})}{2(F_{\text{LC}}P_{\text{HC}} - F_{\text{HC}}P_{\text{LC}}) - P_{\text{Sys}}(F_{\text{HC}} - F_{\text{LC}})} \quad (6)$$

Proof: We prove this for two cases: $\Delta \leq \Delta_{\text{opt}}$ and $\Delta > \Delta_{\text{opt}}$.

Case 1: $0 \leq \Delta \leq \Delta_{\text{opt}}$. In Scenario 1, $A_{\text{HC}} \leq A_{\text{LC}}$. Therefore, the active time of the system $A_{\text{Sys}} = A_{\text{LC}} = C_{\text{Sm}}/F_{\text{LC}}$. So:

$$\begin{aligned}
E_{\text{Total},1} - E_{\text{Total},2} =& P_{\text{HC}}(C_{\text{Lg}} - C_{\text{Sm}})/F_{\text{HC}} \\
&- (P_{\text{LC}} + P_{\text{Sys}})(C_{\text{Lg}} - C_{\text{Sm}})/F_{\text{LC}}
\end{aligned}$$

The positive term in this equation is never greater than the negative term due to conditions 1 and 3. Therefore $E_{\text{Total},1} \leq E_{\text{Total},2}\ \forall\ 0 \leq \Delta \leq \Delta_{\text{opt}}$.

Case 2: $\Delta > \Delta_{\text{opt}}$. In Scenario 1, $A_{\text{HC}} > A_{\text{LC}}$. Therefore, the active time of the system $A_{\text{Sys}} = A_{\text{HC}} = C_{\text{Lg}}/F_{\text{HC}}$. We now find the condition where $E_{\text{Total},1} - E_{\text{Total},2} \leq 0$

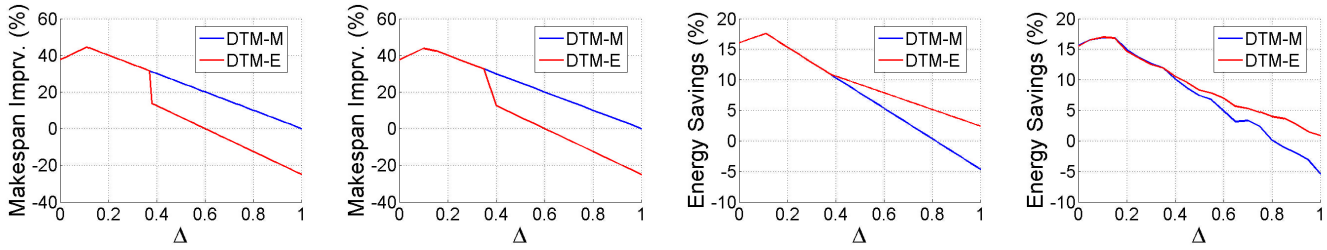
$$\begin{aligned}
&P_{\text{HC}}(C_{\text{Lg}} - C_{\text{Sm}})/F_{\text{HC}} - P_{\text{LC}}(C_{\text{Lg}} - C_{\text{Sm}})/F_{\text{LC}} \\
&\quad - P_{\text{Sys}} \cdot C_{\text{Lg}}(F_{\text{HC}} - F_{\text{LC}})/(F_{\text{HC}} \cdot F_{\text{LC}}) \leq 0 \\
\implies &\frac{P_{\text{HC}}}{F_{\text{HC}}} - \frac{P_{\text{LC}}}{F_{\text{LC}}} - P_{\text{Sys}} \frac{C_{\text{Lg}}}{C_{\text{Lg}} - C_{\text{Sm}}} \frac{F_{\text{HC}} - F_{\text{LC}}}{F_{\text{HC}} \cdot F_{\text{LC}}} \leq 0
\end{aligned}$$

Substituting $\frac{C_{\text{Lg}}}{C_{\text{Lg}} - C_{\text{Sm}}} = \frac{\Delta+1}{2\Delta}$ and solving to Δ leads to the R.H.S of (6). Combining case 1 and case 2 conditions proves the theorem. ■

A. Task Allocation Policies

Firstly, all tasks are partitioned with the objective of making $\Delta = \Delta_{\text{opt}} = \frac{F_{\text{HC}} - F_{\text{LC}}}{F_{\text{HC}} + F_{\text{LC}}}$. We utilize a heuristic algorithm that does bin-packing of tasks to cores with the objective of $\Delta = \Delta_{\text{opt}}$. Specifically, given a set of tasks and two uniform (non-identical) cores, assign tasks to cores such that makespan is minimized. This is a well studied $Q2||C_{\max}$ problem and is known to be NP-Hard. However, the Modified Longest Processing Time (MLPT) heuristic proposed in [12] solves this problem with an approximation ratio of 1.22. MLPT also has low complexity $O(n \log n + c)$ and therefore can be applied at runtime. The working of MLPT algorithm is summarized here for completeness:

1. Sort all tasks in non-increasing order of their computation times.
2. if $n \leq 2$, assign tasks greedily such that their individual completion/finish time is minimized.



(a) Makespan Simulation (b) Makespan Experiment (c) Energy Simulation (d) Energy Experiment

Fig. 3. Dynamic Threshold Mapping (DTM) Energy and Performance Results for $F_{LC} = 80MHz$ and utilization = 50%.

3. if $n \geq 3$, assign the first three (largest) tasks optimally such that their makespan is minimized. Assign the remaining tasks greedily such that their individual completion/finish time is minimized.

Next, the task bins are executed using the following policies:

1. *Delta Threshold Mapping-Makespan (DTM-M)*: Execute *Large* task bin on the HC and *Small* task bin on LC.
2. *Delta Threshold Mapping-Energy (DTM-E)*: Execute *Large* task bin on HC and *Small* task bin on the LC if $\Delta \leq \max\{\Delta_{\text{opt}}, \Delta_{\text{diff}}\}$ **OR** $\frac{C_{\text{Lg}}}{F_{\text{LC}}} \leq D$. Do the converse otherwise.

The optimality of these approaches for a given partitioning has been proved in Theorem 2 and 4.

V. Evaluation Methodology

In this section, we present our experimental results in order to validate our theoretical findings and prove the merits of the Heavy-Light architecture. First, we will use randomized task sets to compare an energy optimal partitioning to state-of-the-art heuristics with and without DTM. Afterwards, we will use fixed partitions to evaluate our DTM algorithms, with both synthetic and real-world benchmarks.

Hardware Testbed: Since the Heavy-Light platform is not commercially available, we can only emulate heavy light behavior using a commercially available dual-core platform. In this work, we use the LPCXpresso54102 platform, which consists of a performance-oriented Cortex M4 core and an energy-efficient Cortex M0 core. The only difference with a Heavy-Light platform is that the LPC has heterogeneous microarchitectures. In order to emulate a Cortex M4-based Heavy-Light platform, we characterize our applications by running them on the M4 at different frequencies, recording their execution times and energy. To test our task allocation policies, the M0 is configured to consume either the same clock cycles or energy as a Light M4 core running a task at a given F_{LC} would, depending on the experiment. Tasks are executed in bare metal and the communication between cores is minimal, so any overheads are negligible when compared to task execution lengths.

TABLE I
Estimated Power Values for Heavy-Light Platform

Component	P_{Active}	P_{Sleep}	$P_{Difference}$
Heavy (100MHz)	22.1 mW	2 mW	20.1 mW
Light (100MHz)	15.5 mW	1.4 mW	14.1 mW
Light (80MHz)	12.1 mW	1.4 mW	10.7 mW
Peripherals	16.6 mW	2 mW	14.6 mW

The makespan and power consumption of the platform is logged using a National Instruments USB-6216 Data Acquisition board. Table I shows the power consumption of the HC and LC at 100MHz. The former is the measured power consumption of the LPC's Cortex M4, and the ladder is the estimate of an M4 LC with reduced power density. The power consumption of the LC scales linearly with its operating frequency, F_{LC}.

The makespan improvement and energy savings from the Heavy-Light Platform are always compared to a Cortex M4 based, single-core platform. To allow a fair comparison, the energy consumption values for single-core were acquired by executing all the tasks on the HC and subtracting the energy consumed by the LC, which is kept in sleep state.

We first compare the performance of DTM-E to a state-of-the-art heuristic called LP+BP. As was previously mentioned in Section II, the heuristic presented in [9] uses a Linear Programming (LP) formulation and a modified Bin Packing (BP) algorithm to reduce the energy consumption in accelerators.

Comparison with LP+BP and Optimal mapping: To compare DTM-E approach, we generated 150000 random task-sets with up to 16 tasks and 100% utilization. *UUniFast* [13] algorithm is used to synthetically generate tasksets with fixed number of tasks. During this comparison, we assumed that $F_{HC} = 100MHz$ and $F_{LC} = 80MHz$, and considered the power consumption of the components as shown in Table I. By applying both the optimal formulation, DTM-E scheme and the LP+BP

algorithm, we estimated the energy savings compared to a single-core using Matlab. The results can be seen in Figure 4. Firstly, note that the performance of DTM-E is near optimal. The mean and second/third quartile values are nearly identical for all simulation points. However, the fourth quartile values are marginally different. While all schemes converge towards 22% energy savings as the number of tasks increase, there is a clear difference for a low number of tasks between LP+BP and Optimal/DTM-E. We notice that although the mean values of all three schemes are similar, for low number of tasks, the worst case for LP+BP (fourth quartile) is significantly lower compared to the worst-case for DTM-E and optimal results. The difference stems from the fact that applications targeted by LP+BP exhibit a high degree of parallelism. In other words, when Δ is close to Δ_{opt}. If two task bins have significantly different sizes (leading to a high Δ), the LP+BP algorithm is expected to have lower energy savings.

Evaluation using Synthetic Benchmarks: The synthetic benchmark evaluation is used to quantify the effects of the single-core utilization and Δ on performance/energy improvement. For each utilization-Δ pair, we run 100 iterations of the application on the hardware test-bed, logging both the makespan and energy consumption of the Heavy-Light and single-core architectures. During these experiments, we also assumed $F_{HC} = 100\text{MHz}$ and $F_{LC} = 80\text{MHz}$.

We first analyze the impact of single-core utilization on energy savings and makespan. In this set of experiments, we evaluate two values of $\Delta \in \{0.2, 0.5\}$. The results for

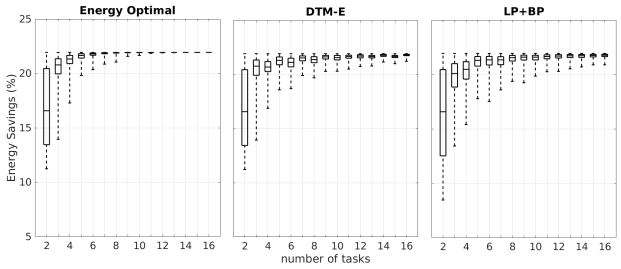
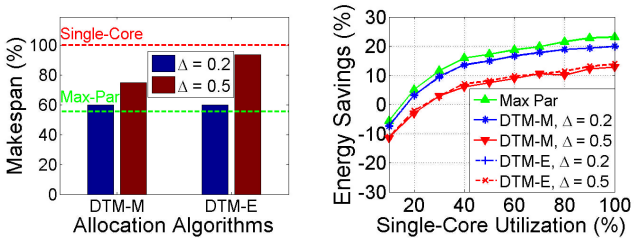


Fig. 4. Optimal vs DTM-E vs LP+BP



(a) Makespan Experiment (b) Energy Experiment

Fig. 5. Makespan and Energy Savings for $F_{LC} = 80MHz$.

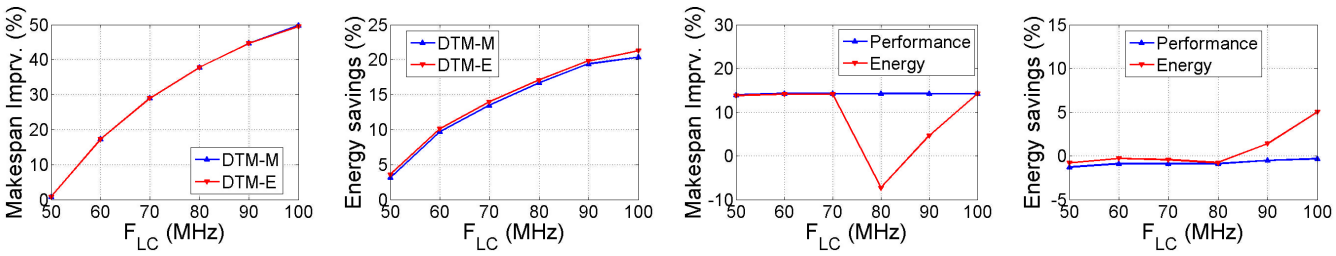
this evaluation are presented in Figure 5. In this case, the % improvement in makespan remains constant as utilization is increased. Therefore, we plot the makespan improvement in the form of a bar graph in Figure 5a. In general, the percentage improvement in energy consumption increases as the utilization increases. We have high energy savings when $\Delta = 0.2$, since it is close to $\Delta_{\text{opt}} = \frac{100-80}{100+80} = 0.11$. Furthermore, the two policies do not differ, as $\Delta \leq \Delta_{\text{diff}} = 0.37$. However, when $\Delta = 0.5$, the two policies differ and there is a tradeoff between performance and energy consumption.

Now, we analyze the proposed schemes for different values of Δ. Figure 6 shows both theoretical results and experimental evaluation for utilization = 50%. According to the plots, the optimal makespan improvement and energy savings are 44% and 17% respectively. Both $\Delta_{\text{opt}}(\approx 0.11)$ and $\Delta_{\text{diff}}(\approx 0.37)$ are in agreement with our theoretical findings. Though the amount of task-level parallelism, and consequently, the potential energy savings are highly application-dependent, we will show in the following section that different types of real-world applications can exhibit improvements in both energy and performance.

Evaluation using Real-World Application: In this evaluation, we demonstrate the impact of our policies on a real-world application. We consider two different embedded applications: 1) Pix4Flow [14] camera module, 2) open-source Inertial Navigation System (INS) [15]. Pix4Flow has high task level parallelism while INS has low task-level parallelism. Though the INS benchmark has dependencies between some tasks, we ignore them to determine the partitioning and respect them when executing the application. During these experiments, F_{LC} is varied from $50MHz$ to $100MHz$ in steps of $10MHz$.

As shown in Figures 6a and 6b, in the case of the Pix4Flow benchmark, the differences between the two task allocation policies are very narrow. This is due to the high degree of task-level parallelism in this application. Interestingly, even in the pessimistic case of $F_{LC} = 50MHz$, both policies offer approximately 4% energy savings. Both makespan improvement and energy savings increase proportionally to F_{LC} and can reach up to 50% and 22% respectively.

For the INS application, the two allocation policies show a very different behavior. This is shown in Figures 6c and 6d. The difference is due to the low task-level parallelism present in the application. When $F_{LC} \geq 80MHz$, the two policies start differing. The energy-efficient policy shows a small improvement in energy consumption (up to 5%). However, the improvement in energy comes at the cost of loss in performance; with makespan degradation of up to 8% compared to single-core due to the LC executing the bigger task bin. The performance-oriented policy has a constant makespan improvement of approximately 14% with a slight increase in energy consumption, since the larger task bin is always executed by the HC.



(a) Makespan Improvement (b) Energy Savings (c) Makespan-INS (d) Energy Savings-INS

Fig. 6. Results for Pix4Flow and INS Benchmarks

VI. Conclusion

In this work, we present a task allocation scheme which targets makespan and energy minimization, for the recently proposed Heavy-Light platform. We prove that only if the level is task-level parallelism is high enough, are the two objectives of minimizing makespan and minimizing energy equal. For scenarios where task level parallelism is low, we propose two Delta Threshold Mapping (DTM) policies which target performance (DTM-M) and energy (DTM-E). We theoretically prove the optimality of the proposed allocation policies and evaluate them on a hardware testbed. The results show an improvement of up to 50% in makespan and up to 22% in energy consumption; compared to a single-core platform.

References

[1] P. Greenhalgh. big.little processing with arm cortex-a15 & cortex-a7. *ARM Ltd.*, 2011.

[2] A. Fuks. Sensor-hub sweet-spot analysis for ultra-low-power always-on operation. In *Symposium on VLSI Circuits*, June 2015.

[3] A. Gomez, et al. Reducing energy consumption in microcontroller-based platforms with low design margin co-processors. In *Proc. DATE Conf.*, 2015.

[4] F. Yao, et al. A scheduling model for reduced cpu energy. In *Proc. Foundations of Computer Science*, pages 374–382, 1995.

[5] H. Aydin, et al. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proc. RTSS*, 2001.

[6] J.J. Chen, et al. Energy-efficient real-time task scheduling in multiprocessor dvs systems. In *Proc. ASP-DAC Conf.*, pages 342–349, 2007.

[7] S. Sarma, et al. Smartbalance: A sensing-driven linux load balancer for energy efficiency of heterogeneous mpsocs. In *Proc. DAC Conf.*, 2015.

[8] M. Kim, et al. Utilization-aware load balancing for the energy efficient operation of the big.little processor. In *Proc. DATE Conf.*, 2014.

[9] F. Paterna, et al. Variability-aware task allocation for energy-efficient quality of service provisioning in embedded streaming multimedia applications. *IEEE Trans. on Computers*, 61(7), 2012.

[10] A. Raghavan, et al. Computational sprinting. In *Proc. HPCA Conf.*, pages 1–12, 2012.

[11] M. Salehi et al. A Hardware Platform for Evaluating Low-Energy Multiprocessor Embedded Systems Based on COTS Devices. *IEEE Trans. on Ind. Electr.*, 62(2):1262–1269, 2015.

[12] C. Koulamas et al. A modified lpt algorithm for the two uniform parallel machine makespan minimization problem. *European Journal of Operational Research*, 196(1):61–68, 2009.

[13] E. Bini et al. Measuring the performance of schedulability tests. *Real-Time Syst.*, 30(1-2), 2005.

[14] Pix4flow website. [accessed 15-No-2015], https://pixhawk.org/modules/px4flow.

[15] J.O. Nilsson, et al. Foot-mounted ins for everybody - an open-source embedded implementation. In *Proc. PLANS.*, 2012.