H-EARtH: Heterogeneous Platform Energy Management

Efraim Rotem[1,2], Ran Ginosar[2], Uri C. Weiser[2], and Avi Mendelson[2]

Abstract—The Heterogeneous EARtH algorithm aim at finding the optimal platform energy point of a heterogeneous cores CPU by selecting the right core and changing the voltage and frequency of operation. The algorithm is based on a theoretical model employing a small number of parameters, which are extracted from real systems using off-line and run-time methods. The model and algorithm have been validated using a cycle accurate simulation, and on real systems using 45nm, 32nm and 22nm Intel® Core processors. The Heterogeneous EARtH algorithm can save an average of 21% energy with up to 33% savings, compared with symmetric core architecture, and up to 44% compared with the commonly used fixed frequency policies in the heterogeneous CPU.

1. Introduction

Energy consumption of modern compute platforms has become a major concern with the growth in data center and client computers deployment. DVFS (Dynamic Voltage and Frequency Scaling) is the most effective method for achieving the best performance for a given power budget by controlling the voltage and frequency of the CPU.

Existing systems use demand based algorithms [1] or Service Level Agreement (SLA) control methods in data centers [2] in order to minimize energy consumption. Recently, new multicore based products introduce heterogeneous core architectures that combine fast, high power cores with slower, power efficient cores aiming at even better energy efficiency. Such architectures include either *asymmetric cores,* sharing the same micro-architecture possibly using different process and design targets [3], [4], or *heterogeneous cores* which use different micro-architectures targeting different energy efficiency levels [5]. These architectures use low power cores when energy efficiency is required and use the big core for high performance operations. Indeed all the above methods assume that smaller core or lower DVFS points are more energy efficient. However, controlling CPU power has limited impact on the overall energy efficiency of the computing platform due to energy consumption of other platform components: While lowering the core's voltage and frequency, or using a power efficient core, decreases core power and energy, computation time is lengthened, resulting in an increase of energy consumed by other platform components [17][18]. When that energy is significant, an alternative policy, Race to Halt (RtH) [7][17], has been proposed where the CPU is operated at its maximum performance point in order to complete computation as soon as possible and turn off the entire platform. Alternatively, when the platform and CPU energy are more balanced, a mid-point over the DVFS scale may result in lower total energy, as shown by the EARtH algorithm [18] and as demonstrated in Fig. 1. In this paper, these findings are extended to heterogeneous multicores.

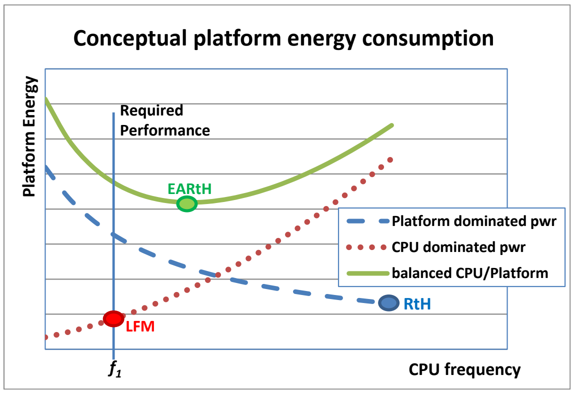

Figure 1: Conceptual total energy in three different platforms. The minimum energy point depends on which portion of the platform dominates power consumption: LFM for CPU dominance, max frequency when rest of platform dominates, and EARtH point when power is balanced. [18]

Fig. 1 [18] exemplifies energy consumption of a platform as a function CPU voltage and frequency. The target performance is achievable by operating at f_1 or faster. When the CPU power dominates total power, the energy follows the dotted curve, and minimum energy is achieved when the CPU operates at f_1. Indeed, this model is assumed in many existing designs [1] and studies [6]. Using a power efficient core further reduces the total energy consumption. More recently, when power dissipation in the rest of the platform has been considered, it has been realized that when the rest of the platform consumes significantly high power compared to the CPU, platform energy follows the dashed curve, and the most energy efficient policy is Race to Halt (RtH) [7][17]. In such a case, the use of a slower, power efficient core might not be desired. In many practical systems, however, power is balanced between CPU and the rest of the platform and energy is represented by the solid curve in Fig 1 above. Previous work [18] has shown that in a homogeneous system, the minimum energy point may happen at some intermediate frequency. Furthermore, [18] has introduced the run-time algo-

1. *Intel corporation*
2. *Technion – Israeli institute of technology*

rithm EARtH to identify and track that minimum energy working point. As we extend the study to heterogeneous systems, it is not obvious whether the fast, high power core, or the slow, energy efficient core, will result in the best platform energy consumption. This paper demonstrates this observation for the first time and presents a novel heterogeneous Energy Aware Race to Halt (H-EARtH) algorithm that identifies the minimum energy point at run time for heterogeneous CPUs. It identifies at run time which core to use and at what frequency, in order to achieve the global minimum platform energy. The paper also validates the H-EARtH algorithm by measurements on real platforms and by cycle accurate simulations.

When considering a minimum energy point, relevant factors that affect power and performance should be accounted for. The relation of frequency to performance or overall execution time depends on parameters such as CPU and platform architecture, workload-dependent memory access patterns and memory organization. Core micro-architecture in a heterogeneous CPU greatly affects the power and the overall workload run time. In this research we evaluate both asymmetric and heterogeneous CPUs. Other relevant parameters include platform and CPU power as functions of workload, core type and voltage frequency operating point. The H-EARtH algorithm presented in this paper accounts for all these parameters, and the paper demonstrates collecting them on real platforms and cross predicting the parameters from the active core to a different type of non-active core at run time. To measure asymmetric cores, we instrumented platforms with two types of the Intel® Core i7 processors manufactured on 45, 32 and 22nm processes: A standard voltage CPU used as the high performance core, and an Ultra-Low Voltage (ULV) CPU for the power efficient core. The algorithm was tested using 37 different benchmarks and at different temperatures. To evaluate heterogeneous cores, we used Intel® ATOM™ core, simulated with the unified interconnect and memory hierarchy, for the small, energy efficient core of the heterogeneous CPU. The paper shows that the H-EARtH algorithm achieves the optimal minimum platform energy accuracy of 2.2%. We demonstrate that heterogeneous CPU, which is operated at this optimal H-EARtH point, achieves an average of 21% energy savings with up to 33% savings compared to a homogeneous CPU. The H-EARtH algorithm can save up to 44% energy compared to either of the two fixed frequency policies, Race-to-Halt (RtH) and Lowest-Frequency-Mode (LFM) operating points. The accuracy of the cross prediction of parameter was 0.62%. In a real system, the actual operation point will be at or above that calculated minimum, according to the SLA requirements set by the operating system. Running slower, or with a smaller core is not energy efficient.

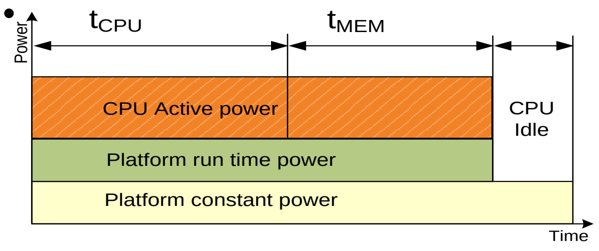

Figure 2: Conceptual platform power over time while CPU is in active and idle states. Platform power is divided into continuous power and power that can be turned off when CPU execution ends.

This paper makes the following observations and contributions:

- An energy efficient core does not always lead to an energy efficient platform. Minimum platform energy may be achieved at an intermediate processor frequency on either a big core or a small core.
- A heterogeneous CPU offers an energy efficient platform, but achieving this energy efficiency requires selecting the proper core and frequency.
- An analytical model identifies the most energy efficient core and calculates the minimum energy frequency, using a small number of parameters.
- H-EARtH algorithm finds the optimal energy point (core type and frequency) in real platforms at run time. The algorithm is based on the required CPU and platform parameters, some produced offline and others collected at run time.
- It is possible to accurately cross-predict the non-active core parameters from a different type of active core at run time.
- H-EARtH algorithm and the model are evaluated on different core designs, process generations and micro-architectures. The model predictions are validated on real platforms and by cycle accurate simulations.

2. The Theoretical Model

We first review the platform energy model of homogeneous cores [18] in Sect. 2.1 and extend it to heterogeneous cores in Sect. 2.2.

2.1 Homogeneous core

A workload run can be characterized as two distinct phases, active and idle, as described in Fig 2. The active phase is further split into interleaved off-chip memory-bound intervals (t_{MEM}) and CPU-bound intervals (t_{CPU}) [9],[10], [11]. While changing the CPU frequency changes the CPU run-time inversely proportional to the frequency, off-chip memory-bound intervals are not affected by CPU frequency. Rather than measuring the time intervals directly, we used the method described below. Furthermore, the energy efficient cores utilize a simpler mechanism to overcome the memory wall and therefore have a higher t_C and same t_{MEM} using the same interconnect.

At first glance the power and energy consumption of modern platforms as a function of CPU frequency seem hard to predict. However, once the different power and energy components are properly categorized, order emerges. We categorize power dissipation into the following components:

- CPU power (dashed Orange in Fig. 2), consumed at run time, comprising both dynamic and leakage parts, having nonlinear dependency on frequency and voltage.
- Platform active power, dissipated by the platform as a result of workload activity, and can be further divided into two sub categories:
  - Fixed energy (not shown in Fig 2): During workload execution, a fixed amount of data is transferred to and from memory, disk drives, etc. If spread over longer time, the power is lower and vice versa but the energy for each transaction is constant. This activity is a function of the application foot print in memory and disk and does not depend on CPU frequency and therefore translates to fixed energy.
  - Constant runtime power (solid Green in Fig. 2): Memory and peripheral devices may consume power as long as there is activity in the system. That power can be turned off during platform idle times. The energy impact of this power is proportional to the run time of the workload and therefore inversely proportional to CPU frequency.
  - Platform constant power (Light Yellow in Fig. 2) dissipated by the platform regardless of workload activity (display, DDR self-refresh, etc.). Unlike runtime power, it is not turned off. Existing techniques can minimize this portion of the platform power [17].

We look for the minimum of the sum of all these energy components. CPU frequency affects the energy resulting from only CPU power and platform active constant power, and hence the optimization process focuses on them. While other, more complex dependencies exist on the platform, our study shows that those are second order effects and can be ignored by the model with minimal impact on overall accuracy.

The platform energy model is described by the following parameters:

f_0 -	*Reference lowest frequency of the CPU*
f_c -	*Frequency, relative to f_0. f_c = factual / f_0.*
t_c -	*CPU bound run-time at fc. t_{c0} is t_c at f_0*
t_m -	*Memory bound run-time, fixed for all f_c*
P_0 -	*Lowest CPU power consumed at f_0*
P_c -	*CPU power at f_c. Power scales as a function of frequency $P_c = P_0 \cdot F(f_c)$.*
P_l -	*Platform active constant power at f_c.*

Given the above notations, the frequency-dependent part E_f of platform energy is:

(1) $$E_f = (t_c + t_m) \cdot (P_c + P_l) = \left(\frac{t_{c0}}{f_c} + t_m\right) \cdot (P_0 \cdot F(f_c\ ) \ + P_l)$$

For the purpose of optimization it is also more convenient to consider energy relative to the platform energy E_{f0} at the reference point f_0. Dividing equation (1) by the same equation with $t_c = t_{c0}$ yields:

(2) $$\frac{E_f}{E_{f0}} = \frac{\left(\frac{t_{c0}}{f_c}+t_m\right)\cdot(P_{co}\cdot F(f_c\ )+P_l)}{(t_{c0}+t_m)*(P_{co}\ +P_l)} = \left(\frac{t_{c0}}{(t_{c0}+t_m)} \cdot \frac{1}{f_c} + \frac{t_m}{(t_{c0}+t_m)}\right) \cdot \left(\frac{P_{c0}}{(P_{c0}+P_l)} \cdot F(f_c\ ) + \frac{P_l}{(P_{c0}+P_l)}\right)$$

We define two platform and workload terms. One is CPU to Platform Power Ratio (CPR), namely the ratio between CPU power at f_0 and total platform power:

$$CPR = \frac{P_{c0}}{(P_{c0}+P_l)}; \quad \text{Clearly } 1 - CPR = \frac{P_l}{(P_{c0}+P_l)}$$

The CPU power is a function of workload characteristics while the components of platform power that impact CPR are not dependent on the workload. CPR can be calculated as described in Sect. 3. Note that leakage power dependency on temperature is accounted for in this power measurement. CPR→1 implies that the platform power is dominated by CPU power, while CPR→0 implies that the rest of the platform dominates the total platform power; in real platforms, CPR lies in between these two extremes.

The second parameter we define is scalability, the ratio of CPU-bound time to total execution time, computed at f_0. We define workload scalability (SCA) as:

$$SCA = \frac{t_{c0}}{(t_{c0}+t_m)}, \text{ and clearly } 1 - SCA = \frac{t_m}{(t_{c0}+t_m)}$$

SCA is a workload characteristic that represents the performance dependency on CPU frequency. High scalability (SCA→1) indicates that performance is CPU bound and tightly related to frequency, while low scalability (SCA→0) indicates that the performance is memory bound and not impacted by frequency. On modern CPU architectures it is not possible to measure workload time intervals t_c and t_m directly because they are tightly interleaved. SCA, however, can be extracted at run time by collecting execution parameters, as explained in Sect. 3.

The platform energy can now be expressed as:

$$(3)\ \frac{E_f}{E_{f0}} = \left(SCA \cdot \frac{1}{f_c} + 1 - SCA\right) \cdot (CPR \cdot F(f_c) + 1 - CPR)$$

To minimize energy, we need to find the frequency that minimizes Equation (3). This equation implies that the relative platform energy is a function of overall run time (which is inversely related to frequency), of the CPU power (reflected in $F(f_c)$, depending non-linearly on frequency), and of SCA and CPR, characteristics of the platform and the workload. A typical core power is a polynomial function of frequency $P_c \propto f_c^{\alpha}$ with α in the range of 1.5 – 3, but the algorithm applies to any function that properly describes the power to frequency dependency. The platform constant power component P_l does not depend on the workload; it is characterized once for this optimization. The platform components that do not depend on the workload, do not impact the optimal frequency as described above.

2.2 Heterogeneous core

We now extend the model to heterogeneous cores. We focus on multithreaded workloads, although the algorithm applies to single threaded workloads as well (Figure 13 and Figure 14). In our heterogeneous core, at any given time, only one core type is active and we can calculate the CPR and SCA values only for that active core at run time. It is therefore needed to predict the parameters of the non-active core from the active core. The interconnect and the memory architecture of our heterogeneous CPU are shared, and therefore t_m for both big and small cores are equal. We approximate the runtime of the CPU bounded portion on the big *vs.* small core as a fixed ratio: $k \times t_{c\_big} = t_{c\_small}$. Using a fixed k is only an approximation, e.g., if the big core has a bigger floating-point unit, workloads that use it extensively might benefit more than others. A balanced mix of different instructions reduces this variance. We evaluate this model on a real system and using simulations. We have defined above:

$$SCA = \frac{t_{c0}}{(t_{c0}+t_m)}, \text{ and clearly } 1 - SCA = \frac{t_m}{(t_{c0}+t_m)}$$

Dividing the two equations (using indices b for the big core and s for the small one):

$$\frac{t_{cb}}{t_m} = \frac{SCA_b}{1-SCA_b}, \text{ and } \frac{K*t_{cb}}{t_m} = \frac{SCA_s}{1-SCA_s}$$

Finally:

$$(4)\qquad \frac{SCA_s}{1-SCA_s} = \frac{K*SCA_b}{1-SCA_b}$$

Equation (4) provides a function to calculate the scalability of a non-active core based on the measured SCA of the active core at run-time (with a known k).

Equation (3) expresses the energy in relative terms. Note that E_0 of the small core is lower than the big core energy at the same reference frequency. In order to compare the energy, we need to place the energy on a common scale:

$$(5)\qquad \frac{E_{0b}}{E_{0s}} = \frac{P_{0b} * RunTime_b}{P_{0s} * RunTime_s}$$

The power at the reference point is measured at system configuration as described in Sect. 3. We use fixed dynamic power ratio to predict CPU power and extract CPR. To calculate the run time of a workload we now divide 1-SCA of the big core by the small core:

$$(6)\qquad \frac{t_{cs}+t_m}{t_{cb}+t_m} = \frac{RunTime_s}{RunTime_b} = \frac{1-SCA_b}{1-SCA_s}$$

Using (5) and (6) we can compare the energy of big and small cores on a common scale and minimize overall energy using (3). Equations (3), (5) and (6) constitute a theoretical model that allows calculating the global minimum energy operating point for a given workload at runtime. Here, the 'operating point' is the combination of core type, operating voltage and frequency. The rest of the paper demonstrates how to practically implement the theoretical model on real systems, evaluate the accuracy of the implementation and evaluate the energy savings achieved.

3. Heterogeneous H-EARtH Algorithm

We propose the H-EARtH algorithm as described in Fig. 3 below and validate its predictions on real systems. It implements the theoretical model above at runtime. The core selection is performed as follows: the minimum energy frequency is calculated using Equation (3) individually for each core type and brought to a common scale using equations (5) and (6). The lowest energy between the cores is selected. The H-EARtH algorithm requires a one-time characterization

```
H-EARtH Algorithm

// Parameter initialization. Offline characterization at
// system design. Parameters stored in, or loaded by
  BIOS at power up

Get P$l$              // Get Platform Run Time Power
Get α                 // Characterize F(fc); Function can be
                      // polynomial, table or other.

// Run time optimization control

Every time interval {
    For each core {
        Pc = CPU power          // Sample CPU power me-
ter;
                                // internal meter or calcu-
                                lated
         CPR = Pc/(Pc+P$l$)
         Get SCA                // Read CPU monitor or use
                                // collected statistics.
        Fopt=min($\left(SCA \cdot \frac{1}{f_c} + 1 - SCA\right) \cdot$
                $(CPR \cdot f_c^{\propto} + 1 - CPR)$)
              over valid frequencies
        Freq = Get Operating System frequency request
        F(resolved) = max(Fopt, Freq)
    }
Scale energy$\frac{E_f}{E_{f0}}$ to a common reference using equations
(5) and (6). Select the core with minimum energy
}
```

Figure 3: The H-EARtH algorithm

procedure at platform production and a run time module. At system production the CPU and platform power are measured at several frequencies. Based on these measurements $P(f_c)$ for each core type are obtained and stored in a non-volatile memory for future use by the H-EARtH algorithm. In the case of polynomial dependency$P_c \propto f_c^{\alpha}$, only α for each core is stored. Measuring P_l directly is not possible. We use linear extrapolation to zero frequency to calculate P_l. Note that in practical implementations this procedure can be done once for a platform model and a BIOS procedure can update the values based on system actual configuration such as DDR memory type and size. The k ratio between the big and small cores is measures on a single, 100% scalable application. At run time, the H-EARtH algorithm calculates CPR and SCA as follows.

Calculating CPR requires the knowledge of CPU power consumption at runtime. Several methods have been proposed in the past to predict the CPU power based on micro architectural event counters [12],[13],[15]. Furthermore, modern CPUs report this value via a built-in power metering, which is used in this study [14]. CPR is then calculated as:

$$CPR = \frac{P_{c0}}{(P_{c0} + P_l)}$$

Calculating SCA, which represents how performance (application run time) scales with frequency. Scalability depends on memory access patterns. Previous work [9],[10],[11] have used memory access patterns to perform DVFS for power performance optimizations. Furthermore, new CPUs (e.g., Intel® Core™ Sandy Bridge) use memory stalls counters to generate scalability metric [18] and optimize energy consumption in active CPU states; that metric has been used in this study. The SCA value of the non-active core is calculated using equation (4).

The H-EARtH algorithm described in Fig. 3 works as follows: One time setting of the computing platform is required. This setup requires measuring P_l and characterizing each of the cores power as a function of voltage and frequency, and storing the results in a non-volatile memory for future use. At runtime, the H-EARtH algorithm is performed once every time interval and calculates CPR and SCA. In our study we evaluated these parameters every 1mSec and performed voltage and frequency decisions every 10mSec. The H-EARtH algorithm is executed on the currently active core but the CPR, SCA and E_{f0} are calculated for each type of core. The algorithm then searches for f_c that minimizes Equation (3) separately for each of the cores. In our study we used a linear search. There is small number of valid frequency points (8 in our implementation) and therefore the computational overhead is very small. The calculated energy of the cores is compared and the core type that results in minimum energy is selected for the next time interval. Finally, the optimal frequency is combined with the frequency requested by the operating system based on the required level of service. The energy savings results in this paper are measured without any minimum service level requirements. They represent the maximum energy savings potential. Lower energy savings will be achieved if the CPU is driven by the operating system to a higher frequency in order to deliver higher performance.

4. INSIGHTS FROM THE THEORETICAL MODEL

We evaluate Equations (3), (5) and (6) and extract some practical insights. This section is merely a parametric study of the theoretical model. The actual measured results of real workloads are described in Sect. 5. CPR$\rightarrow$1 implies that the platform power is dominated by the CPU and the least frequency mode (LFM) policy is preferable. Furthermore, the smaller, more power efficient core may further improve energy efficiency. For CPR$\rightarrow$0, the power is dominated by the platform and RtH policy on the big core should achieve lower energy consumption; the smaller core typically does not help reduce energy any further—it may actually result in higher energy consumption. While previous works [10],[11],[17] were limited to these two frequency extremes, we extend the analysis to the entire range in between and these extreme points become special cases of the model. Fig. 4 illustrates the relative energy as a function of frequency for different SCA and CPR value.

Fig. 4a describes the relative total energy as a function of CPU frequency and different SCA values, where CPR is fixed in this example at some typical value of 20%. The minimum energy point is marked on each chart by a red dot. A low SCA value (uppermost line chart) implies that the application is mostly memory-bound and therefore increasing the CPU frequency does not reduce the run time significantly. Running the CPU at higher frequency increases the CPU power and energy. The total platform energy however is not be reduced enough to compensate for the higher CPU power. The optimal frequency for low SCA values is therefore as low as possible. For higher SCA values (lower lines on the chart) the run time scales well with CPU frequency and therefore the platform energy savings are higher than energy for workloads with low SCA. Similarly, Fig 4b exemplifies the total energy consumption for different CPR values while keeping SCA fixed at 1. The top chart indicates high CPR, caused by high CPU power relative to the platform. Optimal policy obviously would be running the CPU at lowest frequency and saving power of the highest power component.

As demonstrated in the charts of Fig. 4, the analytical model suggests that there is an optimal frequency point which assures the minimum overall energy for a computational task. This frequency either lies within the operating frequency range, or falls on one of the boundaries. If the minimum resides on the minimum frequency point, the correct policy would be running at LFM. If the minimum resides on the maximum frequency point, the correct policy would be RtH. The optimal operating point is a function of two runtime parameters, SCA and CPR. In this study we find these parameters and select the operating frequency that minimizes platform energy consumption.

Fig. 5 exemplifies the parametric behavior of the H-EARtH algorithm for asymmetric and heterogeneous core CPUs such as [3], [4], [5]. We demonstrate the algorithm such that low power core's highest frequency equals the high performance core's lowest frequency. In this demonstration, selecting the small core results in lower power at every point to the left of the cross-over point in Fig. 5, and on the right of that point the big core leads to lower power. In our actual run time study, this crossover point is a function of the workload behavior and is evaluated at run time. Fig. 5 describes the total platform relative energy and the optimal frequency in similar format to Fig. 4. Observe in Fig. 5a that in several scenarios, the low power core running at its highest frequency provides the lowest platform energy. In Fig. 5b we can see that at low power workloads with high scalability, however, the high performance core provides lower platform energy while high power workloads with low scalability are best run on the low power core and lower frequency.

As described above, the theoretical model supports the claim that the minimum energy can be achieved at either the big or the small core and at an intermediate frequency points. For a heterogeneous core, we observe that the small, low power core is not always the most energy efficient selection for total platform energy management as suggested in prior studies. The right core that minimizes energy consumption for performing a computational task is platform and workload dependent. There is no single policy that can meet all conditions. The H-EARtH algorithm is designed as a runtime tool to calculate this optimal frequency point and select

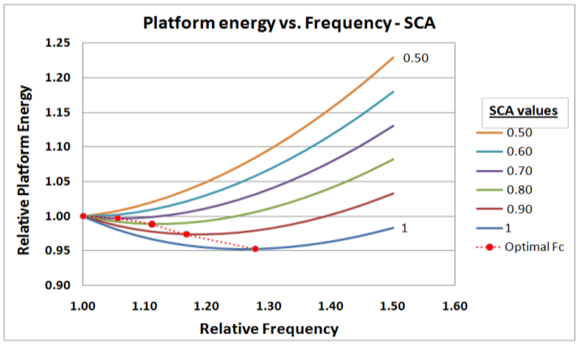

Figure 4a: Modeled platform energy for different SCA values

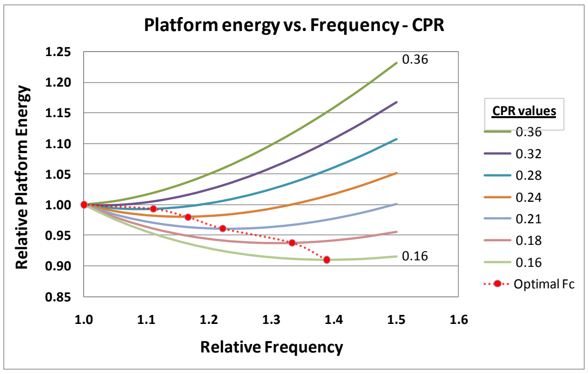

Figure 4b: Modeled platform energy for different CPR values

the right core to perform the computational task at minimum energy consumption. It performs this selection on the fly, and can change voltage, frequency and core selection every time interval, in order to meet changes in execution phases.

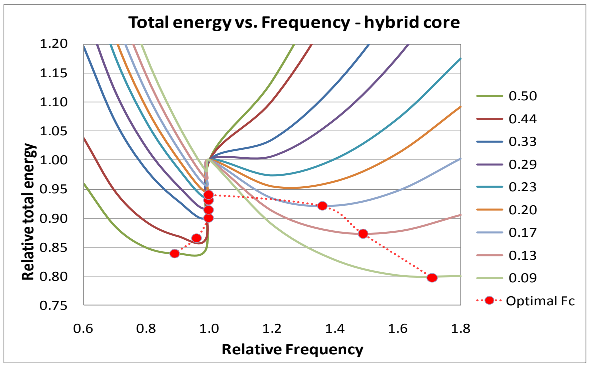

Figure 5a: Platform energy for asymmetric CPU platform for different CPR values with SCA=1

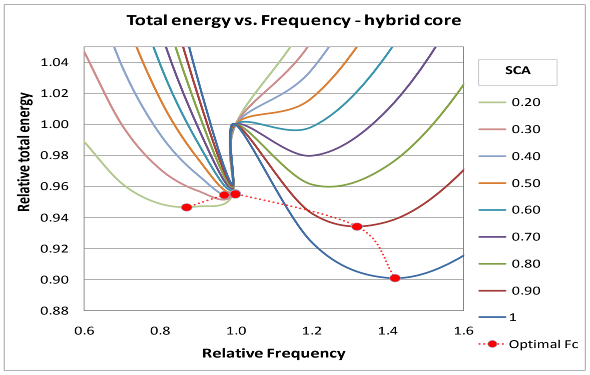

Figure 5b: Platform energy for asymmetric CPU platform for different SCA values with CPR = 0.15

5. Measurements and Simulations

In this section we implement the H-EARtH algorithm on a real system with two core types and also simulate a third core type. The measurements and simulations validate our predictions. In the real system, we implemented a software driver that collected parameters at run time and performed voltage and frequency scaling. We show the accuracy of CPR and SCA. We achieve the minimum energy working point and show significant platform energy savings.

We first describe our measurement system and validate SCA computations and minimum energy predictions in Sect. 5.1. We then describe the measurements in Sect. 5.2 and 5.3, and add simulations in Sect. 5.4.

5.1 Real System Validation

We validate the predictions of the EARtH algorithm on platforms employing state-of-the-art 45nm (Intel® Core™ 2 Duo T9900), 32nm (Intel® Core™ 2 Duo 2860QM) and 22nm (Intel® Core™ 2 Duo 3840QM) processors. Fig. 6 [18] describes measured energy of one workload in a real system that demonstrates the existence of a minimum total energy point in an intermediate frequency.

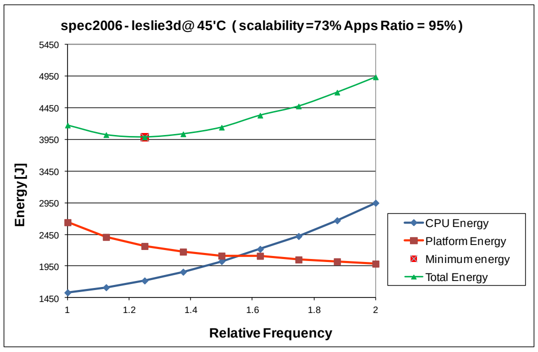

Figure 6: Energy consumption of the CPU, platform and total energy measured on a 32nm Intel® Core™ 2 duo, running SPEC2006. The measurements demonstrate the existence of minimum energy consumption at an intermediate frequency point [18]

We present our measurements on two types of processors: a standard voltage Intel® Core™ 2 Duo 2860QM (measured α~2.4) and an Ultra-Low Voltage Intel® Core™ 2 Duo 2677M (measured α~1.5) intended for Ultrabook computers. The platforms were instrumented to measure the CPU power, various platform components and the total platform power. We used a set of 37 components of Spec-2000, Spec-2006 and SYSmark [16] at two different case temperatures 45°C, 60°C, on the two CPUs at 8 different frequencies.

Validating SCA: We have developed a scalability predictor using a set of micro architectural event counters and collected the scalability value of the selected set of workloads. The actual scalability value is calculated from the workload run time at different frequencies. Fig. 7 compares the predicted and actually measured scalability values. It does not include the training set, used for calibrating the scalability predictor. The accuracy of prediction *vs.* actual value in Fig. 7 is 5.3%.

Platform Energy: The actual measured minimum energy achieved (per workload, temperature and over all frequencies) served as our reference. The H-EARtH algorithm suggested an optimal frequency, and the suggestions resulted in energies that were within 2.2% of our reference.

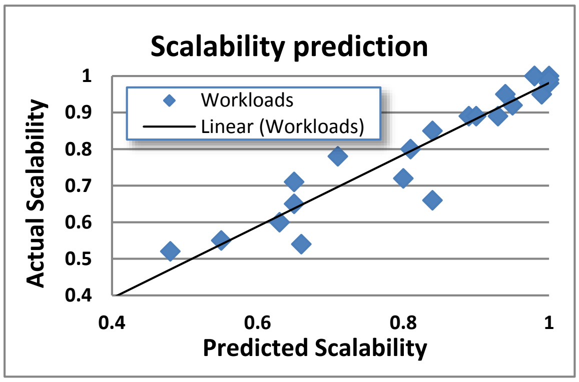

Figure 7: scalability predictor *vs.* measured scalability

5.2 Symmetric CPU

The symmetric core study was done [18] using the run time implementation of the earlier EARtH algorithm. A software driver collected power and scalability metrics and performed DVFS on the system at real time. Data was collected every 1mSec and frequency changes were limited to a single change every 10mSec. The measurements where repeated for each CPU type. Fig. 8.a, 8.b and Tab. I [18] show the potential energy savings of EARtH algorithm compared to two static frequency policies. The horizontal axis lists all benchmark runs, sorted in each chart according to the energy savings level. Evidently, RtH is the better static policy for the low voltage CPU because the power cost of higher frequency is low compared to the rest of the platform. On the other hand, for the standard voltage CPU, LFM rather than RtH is the better static policy, because the CPU consumes higher power. For comparison, a policy that randomly selects the frequency is also shown. While the random frequency policy may save energy relative to one static frequency policy or another, EARtH algorithm outperforms all three policies.

5.3 Asymmetric CPU

In this study, we constructed a model of a CPU consisting of quad high power high frequency cores and quad low power slow cores. Both types of cores have the same microarchitecture and are equal in area. Not having a real CPU with asymmetric cores, the study was performed separately on two CPUs and the results were combined by an offline model. The energy for each of the workloads was measures at 8 different frequencies that were held fixed for the entire workload run. CPU and platform energy were collected and compared with the different policies offline. For each workload, we selected the lowest energy run from all frequencies and

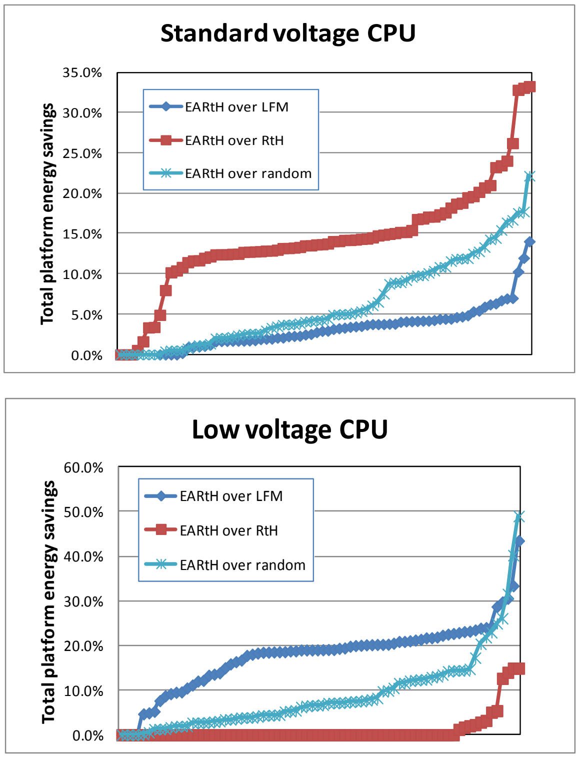

Figure 8: energy savings of EARtH algorithm for (a) low voltage CPU and (b) standard voltage CPU compared to LFM, RtH and random frequency policies [18]

Table I: Average and maximum energy savings of earth compared to a static policy [18]

Energy Savings	Standard CPU (fast)	Low Voltage CPU (Slow)
Average over RtH	15.9%	1.6%
Max over RtH	33.1%	15.2%
Average over LFM	4.8%	18.4%
Max over LFM	17.0%	43.6%

cores, referred to as the “optimal point”. First, we verify the claim that the asymmetric CPU offers better energy efficiency than a symmetric CPU with big cores only. For each workload we compare the global minimum energy consumption point (on either the fast or the slow cores) to the optimum operating point of the fast core. Fig. 9 plots this energy savings for all workloads sorted. Zero in this chart indicates that the optimal operation point is achieved on the fast core and therefore does not benefit from asymmetric CPU. The asymmetric core achieves up to 31% platform energy savings with an average of 13% over the entire workload set.

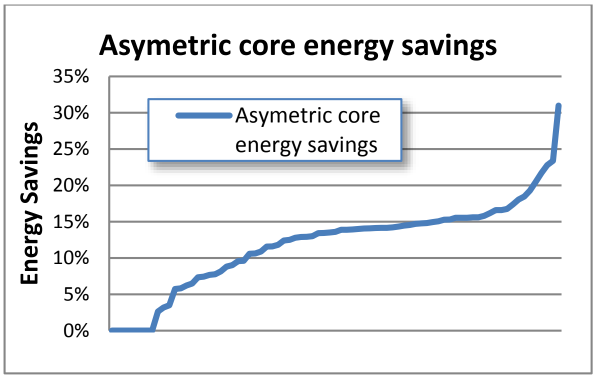

Figure 9: Energy savings of Asymmetric core compared to a CPU consisting fast cores only

Fig. 10 plots the energy savings of the optimal point compared to fixed policies LFM and RtH on each core type. Energy saving is sorted low to high, individually for each core. S in the chart legend and in Tab. II stands for Slow core while F stands for Fast core.

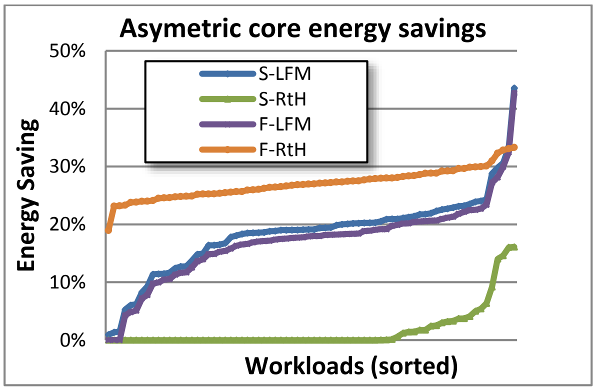

Figure 10: Energy savings of the optimal point compared to fixed policies LFM and RtH on each core type.

Tab. II summarizes the best policy occurrences, i.e., the ratio of workloads achieve minimum platform energy at each policy. H-EARtH in the table indicates an intermediate frequency other than RtH or LFM.

Table II: The number of best policy occurrences that achieve minimum platform energy at each policy

Policy	Occurrences
S-H-EARtH	28%
S-RtH	63%
F-LFM	4%
F-H-EARtH	5%

It can be seen in Fig. 9 and Tab. II that the best fixed policy in most of the workloads is running the small core at its maximum frequency (63% of the workloads). Using this fixed policy, however, results in more than one third of the workloads running at sub-optimal frequency. The H-EARtH algorithm accurately identifies these occurrences and can save up to 16% platform energy (the highest point on the S-RtH chart in Fig. 10). Compared to other fixed policies, H-EARtH algorithm can save up to 44% of platform energy (the highest level in Fig. 10). The slow core is integrated on the CPU in order to save energy and, in most cases, it does. In our study, however, we observed that in 9% of the cases the fast core is more energy efficient than using the slow core (the F- rows in Tab. II).

5.4 Heterogeneous CPU

The heterogeneous core study was performed using a full SoC cycle accurate simulator with power modeling. The model consisted of two 3rd generation Intel® Core™ as the big cores and four ATOM™ small cores (Bay Trail) sharing the same interconnect. We assumed that the area of two small cores equal to the area of one big core. At any one time, either the big cores or the small cores are active (but not both), while the non-active cores are turned off and do not consume power. We simulated a set of multi-threaded SPEC components at the 8 different frequencies and collected power and performance scores. The small cores can run four threads simultaneously while the big cores run two threads. The impact on power and performance for each workload is extracted using the simulator. We use the H-EARtH algorithm to find the optimal frequency that minimizes energy consumption of the entire platform for each workload and for each core type independently. While in the asymmetric CPU study of Sect. 5.3 we allowed changing the frequency every 10mSec, in this simulated study we use a single frequency for the entire run of a workload, as determined by the H-EARtH algorithm using parameter averages. Since the simulator simulates only the CPU, the P*l* parameter for the rest of the platform was adopted from the real system study of Sect. 5.3. We compare the minimum possible energy that can be achieved on any core of the heterogeneous CPU (either the big or the small core) compared to the minimum energy that is achieved on a homogeneous CPU consisting of only big cores. Fig. 11 plots the energy savings of the heterogeneous CPU compared to a homogeneous CPU for all 37 workloads at the two temperatures, sorted in increasing order. The left most 9% of the applications achieve the lowest energy by using the big core (yielding no energy savings on the heterogeneous CPU). The remaining 91% of the applications benefit from the heterogeneous architecture and 31% of them achieve the maximum of 33% energy savings by using the small cores of the heterogeneous CPU. The average energy savings of the heterogeneous CPU over the big core CPU in our system is 21%.

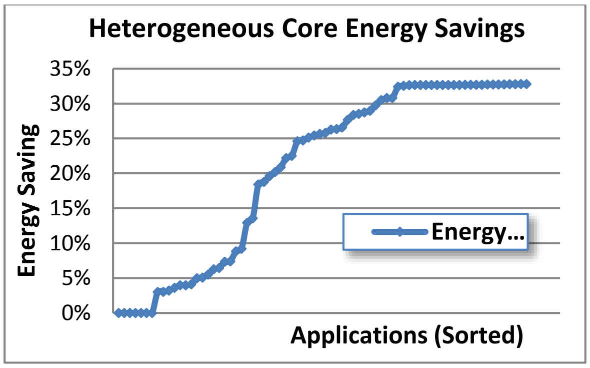

Figure 11: Energy savings of the heterogeneous CPU compared to a big core homogeneous CPU running H-EARtH algorithm

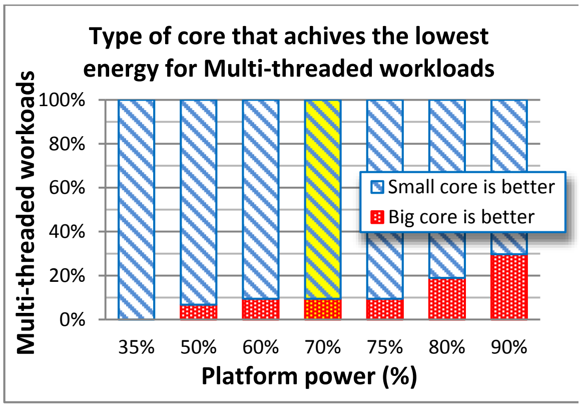

Figure 12 plots the sensitivity of the energy savings to platform power P_l. While in the simulated study we employed specific values of P_l,

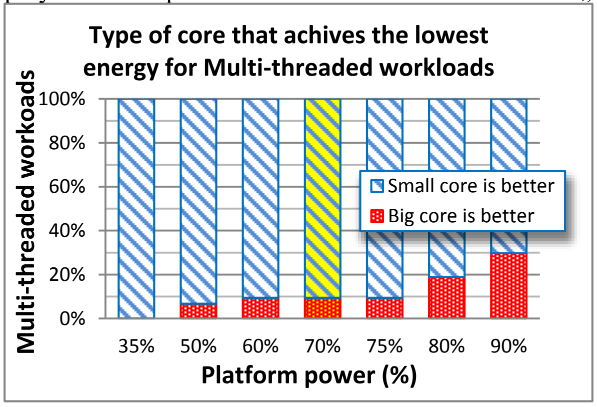

Figure 12 presents other scenarios that may be relevant in other types of platforms. We modified the platform power ratio from 35% to 90% of total power and plot the number of workloads that benefit from heterogeneous CPU compared to big core only. Our real machine platform power is highlights in the chart (70%). As expected, high power platforms benefit more from big core because running fast and going idle minimizes the relatively high energy consumption of the platform. Note: the power of our platform may seem high; this is because the ratio is given at the reference point, with minimum frequency and voltage. Our platform runtime power is only 15% out of the entire power while running the CPU at its highest voltage and frequency, a typical value for Core™ platforms [19].

Single threaded applications utilize only part of the available cores. As a result, the power of fewer cores is smaller compared to the rest of the platform. Furthermore, having several small cores at the same area as a big core benefits only multi-threaded applications. The big core has better single threaded performance than the small core. In this study, all the single threaded applications we tested resulted with lower energy consumption on the big core than the small core. The small core is beneficial only in very low power platforms (Figure 13). Furthermore, Single-threaded workloads achieve the minimum platform energy at a higher frequencies then multi-threaded workloads (Figure 14).

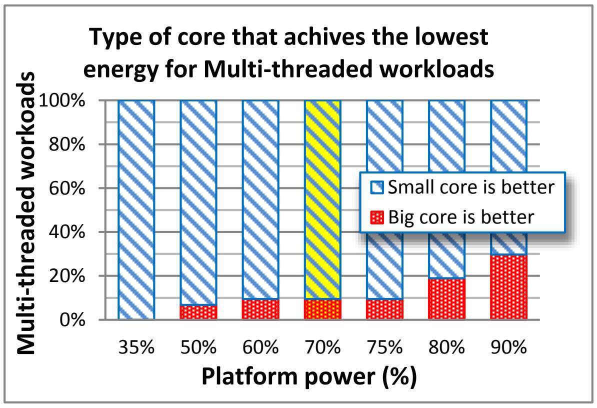

Figure 12: Portion of multi-threaded workload that achieves lower energy on big or small cores, as a function

of platform power

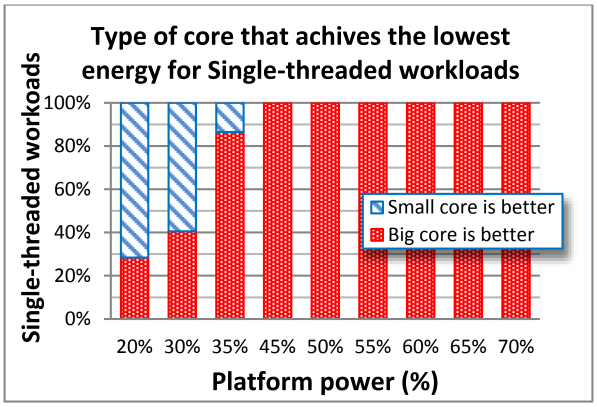

Figure 13: Portion of single-threaded workload that achieves lower energy on big or small cores, as a function of platform power

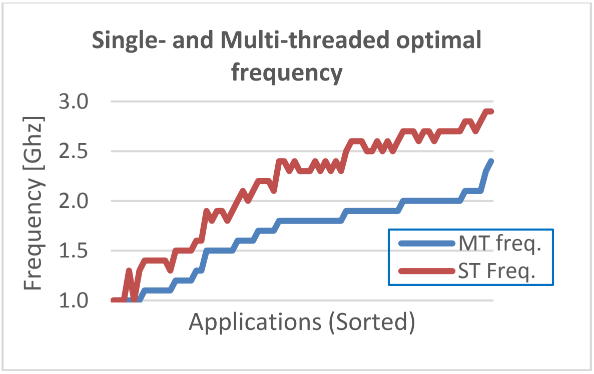

Figure 14: Single- and multi-threaded optimal frequencies. On average, optimal single-threaded frequency is 28% higher than multi-threaded

Extracting SCA of the idle core from the running core: We tested the accuracy of Equation (4) using the simulator. We ran 8 workloads at the 8 different frequencies and measured the SCA for both big and small cores. In our simulation, *k*=1.31. We compared the measured SCA value to the calculated vale. The results are shown in Figure 15. The prediction accuracy average is 0.62% with standard deviation of 0.52%.

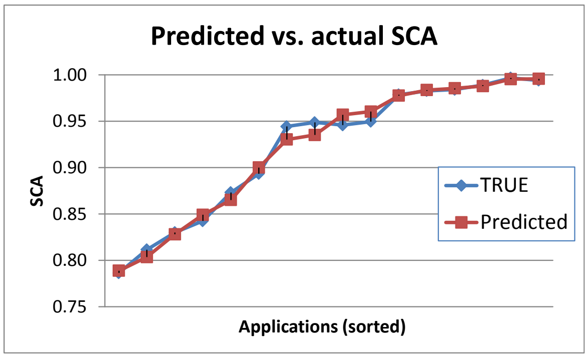

Figure 15: predicted *vs.* measures SCA from the active core type to the idle core and vice versa

6. Related Work

We survey related work in the following four domains. Sect. 6.1 describes previous research on CPU energy conservation methods. Adapting voltage and frequency to minimize platform (rather than mere CPU) power or energy is discussed in Sect. 6.2, and papers on characterizing various parameters on-line are studied in Sect. 6.3. Finally, Sect. 6.4 surveys asymmetric cores.

6.1 CPU energy conservation methods

Prior research considered on-chip *vs.* off-chip activity in the context of DVFS. Hsu & Feng [10] proposed the β adaptive algorithm for effective energy performance tradeoffs. The β value represented on- *vs.* off-chip time. It was used to reduce DVFS frequency at off-chip time intervals in order to trade energy for performance. Kihwan et al. [9] presented a similar concept where off-chip to on-chip ratio was used for fine grain DVFS aiming at optimizing a similar metric of energy to performance tradeoff. Isci et al. [11] proposed a method to predict memory bound phases and reduce voltage and frequency. That method is extended in our study, adding additional micro architectural counters for better prediction of scalability. Furthermore, we show that CPU power (CPR in particular) also needs to be accounted for.

Elyada et al. [20] considered also quality of service considerations in energy-performance tradeoff of DVFS, and proposed a method for energy savings using DVFS while meeting QoS requirement.

Ogras et al. [21] evaluated DVFS in the context of multiple voltage and clock domains in GALS architecture and used the lower voltage and frequency as a means to reduce energy.

Meisner et al. [17] proposed PowerNap method for platform energy savings using RtH policy, finish the work and bring the power of the platform to a very low power as fast as possible. They concluded that in some workload profiles, PowerNap outperforms DVFS methods. In their study, the system idle power was 60%. Modern server platforms such as Intel® Romley have reduced this idle power to as low as 20% [19]. Furthermore, applying the EARtH algorithm to the active portions of such method further improves energy efficiency of the platform [18].

6.2 Voltage and frequency for platform optimal working point

Dhiman at al. [7] evaluated the platform energy consumption at different DVFS policies of a 4 core AMD platform. They also showed that memory access profile of an application affected the energy and performance impacts of DVFS. They showed diminishing energy savings potential of DVFS for the total platform. Dawson-Haggerty et al. [22] evaluated total platform energy of Atom 330 and Intel ® Core 2 Duo and reached a similar conclusion, that the best policy for minimizing energy is ‘Hurry to Sleep.’ The conclusion is based on the observation that there is a fixed component of high platform power.

Each of the above studies considered either one of the two extreme scenarios: a platform that benefits from low frequency or a platform that benefit from a race to halt. Furthermore, no metric has been proposed to conclude which policy is better for a given platform and none of them considered intermediate frequency values. In contrast, this research considers the continuum in between these two extremes, and demonstrates that intermediate scenarios exist. We further evaluate asymmetric cores that extend the CPU power range and provide criteria for selecting the optimal core.

6.3 On line parameters characterization

This paper uses the run time power consumption of the workload and the frequency scalability that is caused by memory access patterns. Various studies have demonstrated the ability to track CPU power at run time. Bellosa [12] demonstrated the capability to predict CPU and DDR memory power at run time by using micro-architectural counters. Contreras & Martonosi [13] performed similar method on a X-Scale platform. Joseph & Martonosi [15] and Isci & Martonosi [23] evaluated power of high performance CPU at runtime. Li et al. [24] studied total platform power. State of the art CPUs such as the Intel® Core™ 2 duo (Sandy Bridge) implement an internal energy monitor that reports accumulated energy and can be accessed at run-time by software [14]. Memory access patterns have been collected and used for power and energy control as described in section 2.1 above. High-end CPUs offer online activity profiling of memory activity and frequency scalability characteristics at run time [14].

Both power characteristics and the memory access profile of the CPU determine the DVFS policy and are used as an input to the EARtH algorithm in this research. We used in this study both offline characterization and online monitoring and compared the results.

6.4 Asymmetric cores

Heterogeneous cores have been studied as means to conserve energy. Kumar et al. [25] have proposed a same ISA, different micro architectures - EV4,5,6 and 8 core. Workloads are scheduled to the different cores according to the workload characteristics and demonstrate significant power savings for small performance penalty. The focus of their work is CPU power and energy with less focus on platform energy. Recently two new asymmetric cores have been introduced. Marvell Armada 628 [4] integrates dual core built with high frequency high leakage process together with a single core manufactured on a low leakage slower technology process. NVIDIA presented Kal-El [3], five ARM Cortex A9 cores, four of which were manufactured on TSMC 40nm general purpose (G) process and operate at 1.4GHz and one core uses low power (LP) process and operates at 500MHz. Arm is offering heterogeneous “big little” core – two micro-architectures with the same architecture as a building block for heterogeneous CPU [3]. Both asymmetric and heterogeneous cores are evaluated in this study.

7. Conclusions

The paper showed that Asymmetric and Heterogeneous CPUs can perform a computational task at lower platform energy than a CPU with only big cores. We demonstrated average energy savings of 21% on all workloads with up to 33% savings on some workloads. The use of small cores, however, is not always energy efficient and an optimal use of the cores depends on platform and workload characteristics. Using the core that is not best suited for the workload and operating it at fixed frequency policy, results in up to 44% platform energy losses. The heterogeneous H-EARtH algorithm achieves the lowest platform energy required to complete a computational task by selecting the right core type and controlling voltage and frequency of that core. We described an analytical model for finding the minimum energy point, based on a small number of physical parameters, collected at production time and at runtime. The analytical model also allows the cross-prediction of non-active cores from the running core. The paper described how to practically implement the analytical model on a real system and extract the required parameters on real platforms. We validated the H-EARtH algorithm by measurements conducted on real platforms with high and low power Intel® Core i7 CPUs manufactured on 45, 32 and 22nm processes. Energy consumption of 37 benchmarks of the SPEC2000, SPEC2006 and SYSmark, at different ambient temperatures, was measured. The heterogeneous CPU was validated using a cycle accurate simulator of ATOM™. We demonstrated the existence of minimum energy consumption point of heterogeneous CPU platforms, and the ability to calculate that point at runtime with accuracy of 2.2%. In conclusion, the H-EARtH algorithm enhances energy-efficient usage of heterogeneous CPUs by enabling runtime core selection and energy management.

8. References

[1] Advanced Configuration and Power Interface (ACPI) Specification, [online], Available: www.acpi.info/

[2] Fan, X., Weber, W., and Barroso, L. A. 2007. Power provisioning for a warehouse-sized computer. In Proc. of the 34th Annual international Symposium on Computer Architecture

[3] Nvidia Kal-El, [online], Available: http://www.nvidia.com/content/PDF/tegra_white_papers/Variable-SMP-A-Multi-Core-CPU-Architecture-for-Low-Power-and-High-Performance_v1.1.pdf

[4] Marvell ARMADA 682, [online], Available: http://www.marvell.com//company/press_kit/assets/Marvell_ARMADA_628_Release_FINAL3.pdf

[5] ARM Big little [online], Available: http://www.arm.com/products/processors/technologies/bigLITTLEprocessing.php

[6] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose and M. Martonosi, “An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget,” In Proc. 39th Annual IEEE/ACM Int. Symp. on Microarchitecture, 2006.

[7] G. Dhiman, K. K. Pusukuri and T. Rosing, “Analysis of Dynamic Voltage Scaling for System Level Energy Management,” Proc. HotPower 08 Workshop Power-Aware Computing and Systems, Dec. 2008.

[8] Patterson, M.K.; , "The effect of data center temperature on energy efficiency," Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008.

[9] C. Kihwan, R. Soma and M. Pedram, “Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 24(1), 18-28. January 2005.

[10] C. Hsu and W. Feng, “Effective dynamic voltage scaling through CPU-boundedness detection,” Proc. 4th Workshop Power-Aware Computer Systems, December 2004.

[11] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. 2006. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In Proc. 39th IEEE/ACM International Symposium on Microarchitecture (MICRO 39).

[12] F. Bellosa, “The benefits of event driven energy accounting in power-sensitive platforms,” Proc. 9th ACM SIGOPS European workshop: beyond the PC: new challenges for the operating platform, Sep. 2000, Kolding, Denmark.

[13] G. Contreras and M. Martonosi, “Power prediction for Intel XScale® processors using performance monitoring unit events,” Proc. Int. Symp. Low Power Electronics and Design, Aug. 2005

[14] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, Doron Rajwan, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," IEEE Micro, vol. 32, no. 2, pp. 20-27, March-April 2012

[15] R. Joseph and M. Martonosi, “Run-time power estimation in high performance microprocessors,” In ISLPED ’01, pages 135–140, 2001.

[16] Standard Performance Evaluation Corporation, [online], www.spec.org/

[17] D. Meisner, B. T. Gold, and T. F. Wenisch. 2009. PowerNap: eliminating server idle power. In Proceedings of the 14th international conference on Architectural support for programming languages and operating systems (ASPLOS '09). ACM, New York, NY, USA

[18] E. Rotem, R. Ginosar, U. C. Weiser, A. Mendelson, "Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management," IEEE Computer Architecture Letters, vol. 99, no. RapidPosts, p. 1, , 2012

[19] Fourth Quarter 2012 SPECpower Results, [online], Available: http://www.spec.org/power_ssj2008/results/res2012q4

[20] A. Elyada, R. Ginosar and U. Weiser, "Low-Complexity Policies for Energy-Performance Tradeoff in Chip-Multi-Processors," Very Large Scale Integration (VLSI) Systems, IEEE Trans. vol.16, no.9, pp.1243-1248, Sept. 2008.

[21] U. Y. Ogras, R. Marculescu, P. Choudhary and D. Marculescu, “ Voltage-frequency island partitioning for GALS-based networks-on-chip,” In Proc. 44th Annual Design Automation Conference, June 2007, San Diego, California.

[22] S. Dawson-Haggerty, A. Krioukov and D. Culler, “Power Optimization – a Reality Check,” Berkeley Technical Report No. UCB/EECS-2009-140, October 19, 2009.

[23] C. Isci and M. Martonosi. 2003. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In Proc. 39th IEEE/ACM International Symposium on Microarchitecture (MICRO 36).

[24] T. Li and L. K. John, “Run-time modeling and estimation of operating platform power consumption,” In Proc. SIGMETRIC ’03, 2003.

[25] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, D. M. Tullsen, "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," In Proc. 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'03), 2003