Evaluating and Modeling Power Consumption of Multi-Core Processors

Robert Basmadjian
University of Passau
Innstrasse 43
Passau, Germany
basmadji@fim.uni-passau.de

Hermann de Meer
University of Passau
Innstrasse 43
Passau, Germany
demeer@fim.uni-passau.de

ABSTRACT

Recently, energy-efficient computing has become a major interest, both in the mobile and IT sectors. With the advent of multi-core processors and their energy-saving mechanisms, there is a necessity to model their power consumption. The existing models for multi-core processors are based on the *assumption* that the power consumption of multiple cores performing parallel computations is equal to the sum of the power of each of those active cores. In this paper, we analyze this assumption and show that it leads to lack of accuracy when applied to modern processors such as quad-core. Based on our analysis, we present a methodology for estimating the power consumption of multi-core processors. Unlike existing models, we take into account resource sharing and power saving mechanisms. We show that our approach provides an accuracy within a maximum error of 5%.

Categories and Subject Descriptors

C.0 [**GENERAL**]: Systems specification methodology; C.4 [**Performance of Systems**]: Design studies, Performance attributes, Modeling techniques

General Terms

Design, Measurement, Performance

Keywords

Component-based modeling, Multi-core processors, Power consumption

1. INTRODUCTION

Lately, the power consumption of processors has become a key concern for energy-efficient computing systems. It was shown in [11, 18, 19] that processors contribute between 23-40% to the total server's power draw. Furthermore, the power drained by a processor mostly depends on its energy-aware mechanisms (e.g. Intel SpeedStep) and load. In 2005, Barroso et al. [5] analyzed Google servers during peak utilization and showed that processors consumed about 57% of the total server's power consumption. However, this percentage in 2007 dropped to 43% thanks to the emergence of energy-aware mechanisms. This variation, which is highly related to the load as well as energy-aware features, demands a thorough understanding of the power consumption behavior in relation to these factors.

Given the importance of the topic, several power consumption models for single- [8] and multi-core [7, 15] processors have been proposed. However, these models have three key limitations: i) they take into account only processors with at most two cores (e.g. dual-core processors), ii) the impact of energy-saving techniques such as Intel SpeedStep [20] and AMD Cool'n'Quiet [1] have not been considered, and iii) for a given same load on cores, it is assumed that the power consumption of each active core is identical due to their similar behavior [23]. Consequently, the overall power consumption of a multi-core processor is considered in the above mentioned models as the sum of power consumption of its constituent cores. However, when several cores are active (e.g. performing computations), they can share resources such as off-chip cache. With such sharing, cores reduce their access to memory. Hence, cores accessing the memory require different power than the ones which do not. As a matter of fact, all the above-mentioned power estimation models suffer from an inaccuracy of up to 62% when I/O bound jobs are executed as described in Section 3.

In this paper, we circumvent the above-mentioned drawbacks by proposing a model that estimates the *dynamic* power consumption of multi-core processors. Due to the variable behavior of the different components of a processor, we decompose the modeling process into the following three component levels: i) processor's chip: these are components located out of dies (e.g. on-chip voltage regulator), ii) die: these are components within a die (e.g. off-chip cache) and iii) core: these are components within a core (e.g. control unit and on-chip cache). For a single core, the power consumption is modeled using capacitance methods [10] based on the core's utilization. When several cores are active, inter-core (cores on the same die) and -die (cores on different dies) communications occur. In this regard, the power consumption of each communication is modeled.

With the emergence of energy-saving mechanisms, the behavior of power dissipation is different than that without such mechanisms. In order to reflect this aspect, we provide a model that estimates the power consumption with and without energy-efficient mechanisms.

The main motivation of this paper is two fold: i) evaluating the power consumption predictions of the previously proposed models [7, 15], and ii) providing a power model for multi-core processors by taking into account the behavior of individual and multiple cores in terms of sharing and energy-saving techniques. To this end, we propose a component-based power consumption model for multi-core processors. This distinguishes our work from the existing models and provides enhanced accuracy. We show that our model provides an accuracy within a maximum error of 5%.

The rest of the paper is structured as: Section 2 gives a brief overview of the related work. The existing models are evaluated in Section 3. Our proposed models are introduced in Section 4. Results are presented in Section 5 to evaluate the developed models. Finally, Section 6 concludes the work.

2. RELATED WORK

A variety of power consumption models both for single- and multi-core processors have been proposed in the literature. For single-core processors, the power consumption is measured directly at hardware-level such as CPU cycles [8], circuit [16] and register-transfer-level (RTL) [9, 14]. The main advantage is that these models provide a high level of accuracy. However, monitoring the activities of a processor at low (transistor) level is complex since a processor has millions (billions) of transistors and monitoring each transistor is not trivial. To overcome this complexity, software-level models have been developed. The power dissipation of the underlying hardware (i.e. CPU) is predicted based on the power consumed by each instruction [22] or function [21] it executes. One key issue is that software-level models depend upon tracing tools that parse an application to determine all its constituent instructions or functions. In case tracing tools are unable to extract the complete information regarding instructions, software-level models suffer from inaccuracy in power estimation.

In order to prevent the dependency on tracing tools, models based on the performance monitoring counters (PMC) [17, 7] have been proposed. Basically, power dissipation during application execution is highly related to the amount of accesses to cache and switching activities within processors. Such activities (events) have been monitored through embedded programmable event counters [6] to calculate the total power consumption of a processor. The major drawback is that certain processors (e.g. AMD Opteron) can report only 4 out of 84 events [24]. Hence, PMCs are unable to provide enough information, and models based on them suffer from inaccuracy for predicting the power consumption.

We overcome these gaps by removing the dependency on the tracing tools, PMCs, and propose a component-based model which demands basic information such as frequency, voltage and load of a core. The major disadvantage of the above-mentioned models is that they do not take into account modern energy-saving techniques. Furthermore, they don't differentiate the variable behavior of cores having parallel or stand-alone computations. To overcome these problems, our model takes into account the behavior of individual and multiple cores as well as energy-saving mechanisms.

3. EVALUATING EXISTING MODELS

The existing models [15, 7] for multi-core processors assume that the overall power of such processors is the sum of power of their constituent cores. Based on this assumption, the overall power of multi-core processors is given by:

$$P_n = \sum_{j=1}^{n} P_c(j), \tag{1}$$

where P_n denotes the power consumption of n cores and $P_c(j)$ represents the power dissipation of a core j.

The key concern is that such models assume that the power consumption behavior of a core remains identical regardless it performs computations either (1) alone and the others stay idle or (2) in parallel with other cores. This assumption is considered due to the similar behavior of cores [23], which is not always adequate. One major counterexample is the sharing of resources. For instance, when several cores share off-chip cache and one core fetches data from the main memory (RAM), the others may not need to further access the memory, if the required data has already been extracted. Instead, they can fetch data directly from the cache. On the other hand, if each single core performs computation alone and the others remain idle, each such core has to access the memory. In other words, the frequency of accessing memory decreases due to sharing. Consequently, the power consumption of several cores becomes less than the sum of their individual powers as given in Equation (2).

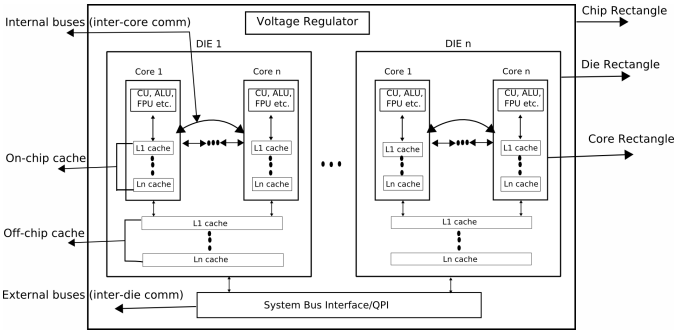


Figure 1: An abstract architecture of a multi-core processor.

In order to analyze the impact of sharing resources, mandatory, exclusive and optional components have been first identified and then investigated. Figure 1 shows an abstract architecture of multi-core processors, which may consist of several dies, and each one can have several cores. Components within multi-core processors can be lied within chip-, die- and core-level as illustrated in Figure 1 in the form of enclosed rectangles. All the components that lie within core-level rectangle are limited (exclusive) to a specific core, and cannot be shared with other cores. Components outside the core-level rectangle are the non-exclusive ones, which can be shared between cores and dies. Some shared components are mandatory, which can be at chip-level (e.g. on-chip voltage regulator) and at die-level. On the other hand, some shared components (e.g. off-chip cache) can be optional. With these aspects, the most relevant components of a multi-core processor can be classified into three generic categories: i) mandatory components (chip- and die-level), ii) exclusive components and iii) optional components. In this section, Equation (2) is evaluated from the perspective of above three components' categories.

Power Component	Benchmark	Operation Type
Mandatory, ALU	INT	Arithmetic operation
FPU	FLOAT	Floating point operation
Control Unit	CU	only instruction execution
On/Off-chip Cache, Buses	I/O	read/write from/to cache

Table 1: Micro-benchmark

3.1 Evaluation Methodology

Equation (2) is evaluated from two different perspectives: i) without energy-efficient mechanisms (EEMs) and ii) with energy-efficient mechanisms. Unlike the latter case, for the former, all the energy-saving features (e.g. DVFS) have been disabled through BIOS Phoenix utility. All experiments provide average power consumption over a period of 15 minutes obtained through a power meter [2] directly monitoring the processors 12V channel. The power consumption is computed based on the following methodology:

1. The power consumption of an individual core j ($P_c(j)$) is first measured, when it performs computations and the others remain idle. To compute power of n cores (P_n), $P_c(j)$ is added n times as in Equation (2). We present such a power as *additive power consumption (Add)* of n cores.

2. To compute the actual power consumption of n cores, the power of these cores is measured when they carry out computations simultaneously. We denote such a power as *actual power consumption (Act)* of n cores.

Note that *Add* and *Act* indicate only the dynamic power consumptions and are computed by subtracting the idle power consumption from the total one. The evaluation is based on a reference scenario of Intel Xeon Quad-core L5420 processor [3], having a maximum frequency of 2.5 GHz, two P-States (2.0 GHz and 2.5 GHz) and supports C-State (C1E).

In order to carry out our experiments, a set of custom micro-benchmarks is designed for the corresponding components as shown in Table 1. The mentioned class of components are stressed using two types of jobs: i) CPU-bound, and ii) I/O-bound. The former type is used for mandatory and exclusive components, while the latter is used to stress optional components, particularly cache and buses.

3.2 Impact of Mandatory Components

The impact of mandatory components is first analyzed at two levels: i) chip-level and ii) die-level.

3.2.1 Chip-Level Components

In this section, the impact of only those mandatory components (e.g. voltage regulator) is analyzed, which lie outside the die-level rectangle of Figure 1. The impact of these components can be identified when cores on different dies perform computations and do not share optional components such as buses and on-chip cache. To this end, INT benchmark is executed that stresses with increments of 20% load a single core on two different dies. The optional components are accessed only in the beginning. After one second, cores never access these resources anymore. To prevent the impact of optional components (e.g. on-chip cache), we run the experiments for 15 minutes and then start to compute the average power over the next 15 minutes. In this way, only the mandatory components that lie out of die-level and

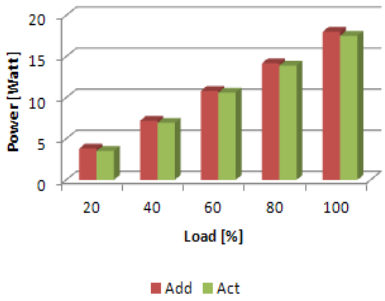


Figure 2: Power consumption of Chip-Level mandatory components.

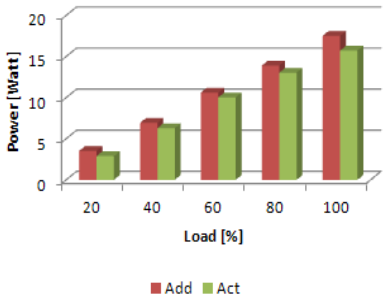


Figure 3: Power consumption of Die-Level mandatory components.

their impact can be detected as shown in Figure 2. It can be observed that the additive power (Add) is always greater than the actual power (Act). However, the difference is not that much significant. Since this difference is at chip-level components, we compute their power consumption by:

$$P_{mc} = Add - Act. \quad (2)$$

From the results of Figure 2, we can conclude that such components have a constant power consumption regardless of whether one or more cores (dies) are active or not.

3.2.2 Die-Level Components

This section investigates the mandatory components that lie within a die-level rectangle and yet outside the core-level one of Figure 1. In this regard, INT benchmark is used for two cores on the same die. The methodology to prevent the impact of optional components is the same as the one for chip-level components. Since chip-level components can not be switched off, these experiments also encompass their impact. In order to identify the impact of die-level mandatory components, the following methodology is adopted:

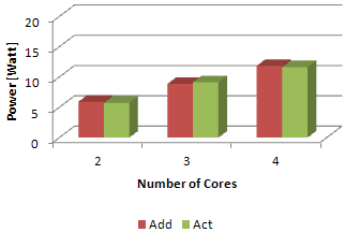


Figure 4: Power consumption of Core-Level exclusive components.

1. The power P_{mc} of chip-level components is measured as in Section 3.2.1.
2. The power P_{mc} in step 1 is subtracted from the additive and actual power consumptions.
3. The resultant additive and actual powers in step 2 indicate the power consumption without chip-level components.

Figure 3 shows the impact of die-level mandatory components. It can be noticed that the additive power (Add) is always greater than the actual power (Act). In contrast to chip-level components, this difference is not negligible. Since this difference is at die-level mandatory components, the reduction in the actual power consumption is considered as the power reduction P_{md}, which is due to such components.

3.3 Impact of Exclusive Components

By exclusive components we mean those which are limited only to a specific core. Since the number of such components (e.g. ALU, FPU) increases as a function of active cores, it is expected that the power consumption of exclusive components also changes with the number of cores. Hence, Equation (2) would hold true in this context. In order to investigate this argument, INT benchmark is used. Since mandatory components always involve, their impact is also included. To exclude this impact, the following methodology is adopted:

1. The power consumption P_{core} of a core is measured, which includes the impact of both mandatory and exclusive components.
2. The power consumption of mandatory component is measured ($P_{md} + P_{mc}$) as in the previous sections.
3. The power of only exclusive component is derived by: $P_{ex} = P_{core} - P_{md} - P_{mc}$.

The results in Figure 4 show that the additive and actual power consumption of exclusive components is identical. Note that other exclusive components such as Floating Point Unit (FPU) have also similar results. Hence, for space considerations, those results are not presented in this section.

3.4 Impact of Optional Components

Since buses and off-chip cache are the most dominant optional components, these were stressed using the I/O benchmark. Each core reads (writes) 64 MB array and does not perform any other operation. The decoupling of cache and

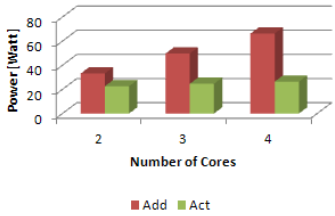


Figure 5: Impact of optional components

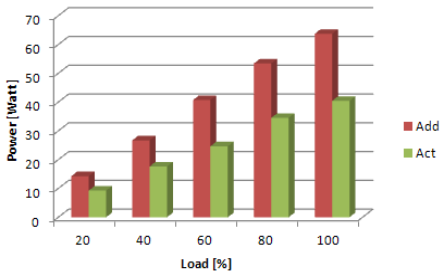


Figure 6: The impact of C-States with CPU-bound jobs.

buses is very complex since reading (writing) data from (to) cache always happens through buses. Hence, it is assumed that the impact of both components can be determined by reading (writing) from (to) cache. In order to compute the power of off-chip cache, we adopt the following strategy:

- The power consumption of a core is computed by excluding optional components (e.g. reading (writing) only from (to) on-chip cache).
- The power consumption of a core is computed by including optional components (e.g. reading (writing) from (to) off- and on-chip caches).
- The difference between both powers is considered as the power of the optional components.

In order to observe the maximum impact of cache sharing, all cores execute same instructions and update the same memory location. Figure 5 illustrates the additive and actual power of cores when they access cache. It can be observed that the actual power of cores is 31-62% less than the additive one. These results show that Equation (2) leads to significant inaccuracy for I/O bound jobs.

We can also notice in Figure 5 that the increase in the actual power consumption (Act) with increasing the number of cores is not significant. We believe the major reason is that all cores have common memory controller, which reads (writes) data from (to) RAM. Since a memory controller (lies on the north bridge of micro-processor) is shared among cores, it has a constant power consumption regardless of whether one or more cores are involved in the computations.

3.5 Impact of Energy-Efficient Mechanisms

Energy-efficient mechanisms (EEM) such as C-States [13] and Intel SpeedStep [20] have been developed to reduce respectively the idle and dynamic power consumption of modern processors. In this section, both aspects are investigated.

	Without EEM		C-State		Intel SpeedStep	
Load (%)	P_{dyn}	P_{tot}	P_{dyn}	P_{tot}	P_{dyn}	P_{tot}
20	6.30 W	23.80 W	9.27 W	18.77 W	7.20 W	16.7 W
40	12.60 W	30.10 W	17.62 W	27.12 W	15.6 W	25.1 W
60	20.53 W	38.02 W	24.64 W	34.14 W	25.5 W	35.0 W
80	26.60 W	44.10 W	34.51 W	44.01 W	32.48 W	41.98 W
100	32.52 W	50.02 W	40.41 W	49.91 W	40.62 W	49.98 W

Table 2: Power consumption comparison for with and without EEMs of Intel Xeon (L5420) Quad-core processor with four active cores.

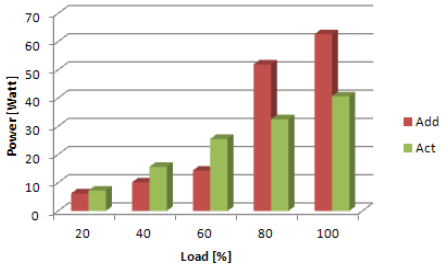


Figure 7: Impact of DVFS with CPU-bound jobs.

Since the mandatory components cannot be disabled, these experiments also encompass their impact.

3.5.1 Idle Energy-Efficient Mechanisms

In these experiments, only idle power related energy-efficient mechanisms (e.g. C-States) are enabled through BIOS Phoenix utility while keeping Intel SpeedStep disabled. The used benchmark is the same as in Section 3.2. The results in Figure 6 show that the decrease in the actual power (*Act*) as compared to the additive power (*Add*) is 33-39%.

3.5.2 Dynamic Energy-Efficient Mechanisms

In contrast to the energy-efficient mechanisms for idle power, these mechanisms reduce the dynamic power consumption through dynamic voltage and frequency scaling (DVFS) [12]. In this section, we analyze the impact of DVFS with Intel SpeedStep using INT benchmark. It can be seen in Figure 7 that for a load[1] less than 60%, *Add* is smaller than *Act*, while the situation is reversed for a load more than 60%. This is due to the variation in the cores' frequency. When only one core is active, the frequency remains at minimum (2.0 GHz in our case) for up to 60% of load[2]. On the other hand, when more than one core perform parallel computations simultaneously, the frequency fluctuates between minimum and maximum (2.0 and 2.5 GHz). Consequently, Equation (2), which sums up the power consumption of one active core with 2.0 GHz four times, underestimates the real consumption of 4 cores performing parallel computations. However, for a load more than 60%, the frequency remains the same (2.5 GHz) both for *Add* and *Act*.

It is worthwhile to note that the dynamic power consumption with EEMs is greater than that without EEMs for the corresponding loads. However, the total power consumption with EEMs is always smaller than without EEMs as shown in Table 2. The main reason is that without EEMs, all components remain switched ON, especially during low utilizations. When a core is loaded, only the dynamic power of the involved component is computed. On the other hand, C-States reduce the idle power by switching off unused components. When a core performs computations, the necessary components have to be switched ON again. Since the power of each component consists of its idle and dynamic ones, the idle power of turned ON components becomes part of the dynamic power.

From these results, it is clear that there is a necessity of a model for multi-core processors, which can reduce such inaccuracy by capturing the behavior of multiple cores in relation with energy-saving mechanisms and resource sharing. In this paper, we propose a model to address these problems, whose details are covered next.

[1]Each of the four cores has the same utilization percentage.
[2]Due to the fact that such a processor has only two P-States.

4. COMPONENT-BASED MODELS

As demonstrated in Section 3, the core architecture of multi-core processors can be categorized into three classes of components: mandatory (chip- and die-level), exclusive (core-level) and optional components. Furthermore, from the analysis presented in Section 3, we can conclude that, unlike certain components (e.g. exclusive), others (e.g. mandatory chip- and die-level) have a constant power consumption regardless of the number of involved active cores.

Given a multi-core processor consisting of multiple dies as shown in Figure (1), inter-die communication may also occur through chip-level components. With all these facts, a model for multi-core processors is given by:

$$P_{proc} = P_{mc} + P_{dies} + P_{int\_die}, \quad (3)$$

where P_{mc}, P_{dies} and P_{int_die} denote respectively the power consumption of chip-level mandatory components, the constituent dies and their inter-die communication. In the rest of this section, we provide models for P_{mc}, P_{int_die}, and P_{dies} respectively.

4.1 Chip-Level Mandatory Components Model

Let P_{mc} indicate the power consumption of chip-level mandatory components, then it can be presented by:

$$P_{mc} = s(v, f), \quad (4)$$

where v and f denote the voltage and frequency respectively, and s represents the capacitance function, which is defined as:

$$s(v, f) = c_{eff} v^2 f, \quad (5)$$

where c_{eff} represents the effective capacitance [10], whose modeling is out of the scope of this paper. Table 3 presents a sample value of P_{mc} capacitance.

4.2 Inter-Die Communication Model

When cores on different dies are active and update data at the same memory location, a communication between cores happens. The power consumption due to the inter-die communication is the sum taken only conditionally over the active dies involved in this communication and is given by:

$$P_{int\_die} = \sum_{k=1 \mid d_k \in D}^{n-1} c_{eff} v_k^2 f_k, \quad (6)$$

where d_k and D indicate respectively the set of active cores of a die k and the set of active dies involved in the communication, which can be determined by:

$$D = \{d_k \mid d_k \neq \emptyset\}, \quad (7)$$

such that

$$d_k = \{c_k(j) \mid l_{c_k(j)} > 0\}, \quad (8)$$

where $l_{c_k(j)}$ denotes the load of the core $c_k(j)$ on the k^{th} die such that $j \in [1, n_k]$ and $k \in [1, n]$ where n_k and n represent respectively the total number of cores of the k^{th} die and the total number of dies of the processor. The voltage v_k and frequency f_k of a die k are the ones of its corresponding cores. Note that when only one core is active, v and f of a core represent v_k and f_k of a die k. If multiple cores on the same die are active, then $v_k = \max\{v_{c_k(j)} \mid l_{c_k(j)} > 0\}$, such that $j \in [1, n_k]$ and $k \in [1, n]$. A similar function is used to compute the frequency f_k. Table 3 presents a sample value of P_{int_die} capacitance.

4.3 Die Level Model

A die in multi-core processors consists of i) die-level mandatory components, ii) cores and iii) off-chip caches. Thus, the power consumption of a die is composed of the above mentioned three parts, which leads to the following model:

$$P_{die}^k = P_{md}^k + P_{cores}^k + P_{off}^k, \quad (9)$$

where P_{die}^k denotes the power consumption of the k^{th} active die such that $k \in [1, n]$ and n indicates the total number of dies of the processor, whereas P_{md}^k, P_{cores}^k and P_{off}^k represent respectively the power consumption of die-level mandatory components, the n_k constituent cores and off-chip cache of the k^{th} die. In case a processor has multiple dies, then we have:

$$P_{dies} = \sum_{k=1 \mid d_k \in D}^{n} P_{die}^k, \quad (10)$$

where P_{die}^k is introduced in Equation (9).

The value of P_{md}^k is computed using the same model as Equation (4) with the corresponding capacitance value given in Table 3, while models for P_{cores}^k and P_{off}^k are given next.

4.3.1 Core-Level Modeling

In modern multi-core processors, more than one core can be integrated on a single die. Thus, we decompose core-level modeling into i) individual, and ii) multiple cores. Due to the fact that several cores on a single die k perform computations simultaneously, then inter-core communications may occur. Hence, the core-level model can be represent by:

$$P_{cores}^k = P_m^k + P_{int\_core}^k, \quad (11)$$

where P_m^k and $P_{int\_core}^k$ denote the respective power consumption of $m = |d_k|$ active cores of a die k and their corresponding inter-core communications.

Individual Cores. As mentioned previously, since cores possess the same exclusive components (e.g. CU, ALU, FPU and on-chip cache) and execute computations in identical manner, the power consumption behavior of each individual core $c_k(j)$ is expected to be similar [23] and is given by:

$$P_{c_k(j)} = P_{exc}^{c_k(j)} + P_{on}, \quad (12)$$

where $P_{exc}^{c_k(j)}$ and P_{on} indicate respectively the power consumption of exclusive components and on-chip cache of a core $c_k(j)$ on the k^{th} die. Note that the power consumption of buses used to access on-chip cache is also entailed inside P_{on} since buses and cache are two inter-dependent components. From Figure 4, it can be seen that exclusive components of cores have a nearly linear behavior in function of their utilization. Based on these results and inspired by [7], we argue that the power consumption of exclusive components increases linearly with increasing the load. Thus, we propose a linear utilization-based model for exclusive components of each individual core, which is given by:

$$P_{exc}^{c_k(j)} = P_{max} \frac{l_{c_k(j)}}{100}, \quad (13)$$

where $l_{c_k(j)}$ denotes the utilization (load) of the core $c_k(j)$ on the k^{th} die, whereas P_{max} indicates the power at maximum utilization which is computed by:

$$P_{max} = c_{eff} v_{max}^2 f_{max}, \quad (14)$$

where c_{eff}, v_{max}, and f_{max} denote respectively the effective capacitance (whose value is provided in Table 3), voltage and frequency at maximum utilization.

On-Chip Cache. Basically, the on-chip cache is also considered as an exclusive component of a core. Since the number of on-chip cache level (e.g. L_1, L_2, etc.) can differ from one core to another, we separate its modeling from the other exclusive components. To model its power consumption behavior, we adopt the following methodology:

1. The power consumption of exclusive components is determined ($P_{exc}^{c_k(j)}$).
2. The total power consumption including on-chip L_i-cache is measured (P_{all}).
3. The power of only accessing L_i-cache is computed by $P_{all} - P_{exc}^{c_k(j)}$.

In order to stress different cache levels L_i, a custom benchmark is developed, which performs read/write operations on the corresponding cache level appropriately. On-chip cache accesses were monitored using the cachegrind tool.

The power consumption P_{L_i} of accessing an on-chip cache L_i is computed using the same capacitance model as Equation (5) with the corresponding value given in Table 3. The total power consumption of all on-chip caches is calculated by summing the corresponding power consumptions of all caches as:

$$P_{on} = \sum_{i=1}^{s} P_{L_i}, \quad (15)$$

Component	Capacitance
Chip-Level Mandatory (mc)	0.103
Die-Level Mandatory (md)	0.301
Without ALU and FPU	1.732
With only ALU	3.124
With only FPU	2.896
On-chip Cache	0.165
Off-chip Cache	3.759
inter-die	0.595

Table 3: Capacitance of different components.

where $s \geq 1$ denotes the core's total number of cache levels.

Multiple Cores. Unlike the existing models [15, 7], the sum of power of cores is valid only for exclusive components of active cores. Thus, the overall power of multiple cores is:

$$P_m^k = \sum_{j=1 \mid c_k(j) \in d_k}^{n_k} P_{c_k(j)}, \tag{16}$$

where $P_{c_k(j)}$ denotes the power consumption of an active core $c_k(j)$ on the k^{th} die given by Equation (12).

Inter-Core Communication. When several cores within a die k are active, inter-core communication may occur. The power consumption of this communication is modeled by:

$$P_{int\_core}^k = \sum_{j=1 \mid c_k(j) \in d_k}^{n_k - 1} s(v, f), \tag{17}$$

where n_k indicates the total number cores of a given die k such that the sum is taken conditionally over the active cores involved in the communication, whereas s is introduced in Equation (5). Since a core can also access optional components such as off-chip cache, its model should also include the power of such components, which is given next.

4.3.2 Off-Chip Cache Modeling

Since off-chip cache, buses and memory controller are interdependent, the decoupling of such components is nearly impossible. Hence, the model regarding the off-chip cache also encompasses the impact of buses and memory controller. To provide a power model of such off-chip cache, the power consumption of each cache level is computed as explained in Section 3.4. For each cache level, the capacitance model is adopted as in Equation (5) and the effective capacitance is given in Table 3. The total power dissipation of all off-chip caches P_{off} is calculated as in Equation (15).

Finally, we can notice from Table 3 that the main contributors in the power consumption of multi-core processors are the processing units (e.g CU, FPU, ALU) and the off-chip cache. Furthermore, it is worthwhile to note that all the values provided in Table 3 cover the case where energy-efficient mechanisms are disabled.

4.4 Energy-Efficient Mechanisms

The energy-efficient mechanisms (e.g. Intel SpeedStep, AMD Cool'n'Quiet, etc.) reduce the power consumption of a multi-core processor by dynamically scaling its voltage and frequency as well as by turning off the unused components. As a matter of fact, the impact of such mechanisms can be reflected indirectly inside our proposed component-based models through the voltage v and frequency f parameters, in addition to excluding the power consumption of those components (e.g mandatory, exclusive, optional) not involved (turned off) during computations.

Since the implementation details of such mechanisms are still out of the public reach, we analyzed them and derived empirical models based on our experiments. We noticed that the power reduction provided by these mechanisms is highly related to the change in frequency f as well as the number of involved active cores. Consequently, we derived power reduction factors by adopting the following methodology:

1. We fixed the frequency to the minimum value f_{min} and performed identical computations on two arbitrary cores. We compared the power consumption of both cores obtained from the power meter with the one obtained by summing up the power consumption of each individual core using Equation (2). Based on this observation, we derived the following first factor:

$$\alpha_{f_{min}} = 1 - \frac{Act}{Add}, \tag{18}$$

where Act and Add denote respectively the power consumption of the processor obtained through the power meter (given by Equation (3)) and Equation (2).

2. To study the impact of frequency f change, we varied it till reaching the maximum value f_{max}. For a given frequency f, we performed identical computations on two arbitrary cores. We compared the power consumptions in the same manner as in step 1. Based on this observation, we derived the following second factor:

$$\beta = \frac{1}{f - f_{min}}((1 - \frac{Act}{Add}) - \alpha_{f_{min}}), \tag{19}$$

where f indicates the given frequency, whereas Act, Add, f_{min} and $\alpha_{f_{min}}$ are given in Equation (18).

3. In order to study the impact of the number of cores on power reduction, we increased the number of active cores for a given frequency f (ranging from minimum to maximum). We compared the power consumptions in the same manner as in step 1. Based on this observation, we derived the following third factor:

$$\gamma = \frac{1}{k}((1 - \frac{Act}{Add}) - \alpha_{f_{min}}), \tag{20}$$

such that $k = w - 2$ where $w > 2$ denotes the total number of active cores of a processor having a load greater than zero and is given by:

$$w = \sum_{k=1}^{n} |d_k|.$$

Based on steps 1, 2, and 3, the power reduction factor is:

$$r = \alpha_{f_{min}} + \beta(f - f_{min}) + \gamma(w - 2), \tag{21}$$

where $\alpha_{f_{min}}$, β, and γ are given by Equations (18), (19), (20) respectively.

Table 4 provides an example of power reduction factor r, computed by means of Equation (21), that can be achieved in quad-core processors using energy-saving mechanisms such as Intel SpeedStep and AMD Cool'n'Quiet. As we can notice, the higher the frequency is, the possibility of reducing

power becomes more evident. Furthermore, more power is saved as the number of active cores increases. These are due to the fact that the pure summation of the power consumption of the cores given by Equation (2) is always greater than the total actual power consumption. As a matter of fact, this difference becomes important for increasing frequencies and number of active (100% loaded) cores. Hence the reduction factor becomes more obvious for those cases. Consequently, the total power consumption of a multi-core processor can be expressed by:

$$P_{proc} = P_n(1 - r), \tag{22}$$

where P_n is introduced in Equation (2) and r is the power reduction factor due to energy-efficient mechanisms.

5. OBTAINED RESULTS

In this section, we present the detailed results of the experiments that were carried out to evaluate the proposed power consumption models.

5.1 Configuration Setup

In our experiments, we used processors from two different manufacturers: Intel and AMD. The list of processors, their corresponding vendors, along with supported frequencies, are presented in Table 5. For monitoring the power consumption of a core (processor), LMG500 [2] power meter is used. The corresponding device takes 10 samples per second. Each observation run takes about 1000 sec. The experiments were carried out on Linux Ubuntu 9.10 OS.

5.2 Benchmark Design

In order to illustrate the accuracy of our proposed models, two strategies were adopted:

1. Equation (3) is evaluated by executing a simple code which comprises only of a *"while(1);"* loop executed on each core individually. In this way, we ensure that minimum number of components (mandatory chip and die, exclusive core – Control Unit) involve in power consumption computations.

2. *lookbusy* [4] benchmark is used in order to compare and evaluate Equations (2) and (22). It imposes synthetic loads on several cores simultaneously.

5.3 Experiment 1

The corresponding experiment was carried out using a simple *while-loop* executed on Intel Xeon quad-core [3] processor having a voltage of 1.1V and a frequency of 2.5 GHz.

Figure 8 illustrates the power consumption obtained through the power meter (*Act*) and through Equation (3) (*Est*). We can notice that both *Act* and *Est* have quite identical power consumptions. Note that during the execution of such a code, only mandatory chip (0.3 W), die (0.9 W) components and the Control Unit of each core (5.25 W) are involved.

5.4 Experiment 2

In order to validate our model introduced in Equation (22), we first show the inaccuracy obtained when using the existing model of Equation (2). Then, we illustrate how this inaccuracy can be reduced by using Equation (22). In both cases, *lookbusy* benchmark is used as a workload generator.

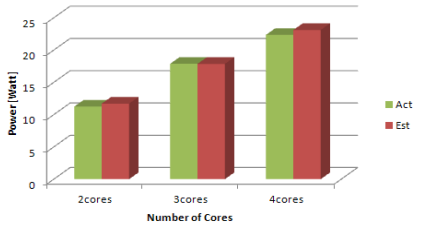


Figure 8: Power consumption of the cores each executing a while-loop.

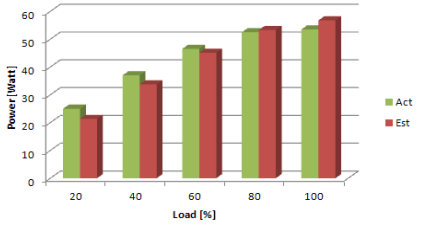


Figure 9: Power consumption comparison of Intel Xeon L5420 quad-core for 4 cores.

Figures 10 and 11 depict the error percentage as a function of the core load. The error rate increases gradually with increasing frequency and number of cores, which confirms our model given in Equation (21). For Intel processor in Figure 10, the maximum inaccuracy can reach up to 33% while for AMD in Figure 11, this error decreases up to 15%. This is due to the variable behavior of energy-saving mechanisms for different manufacturers. However, this inaccuracy can be reduced by applying our model using Equation (22). Due to the variable impact of *reduction factors* for Intel and AMD, different parameters were proposed for Equation (21). For Intel processors, the observed values are: $\alpha_{f_{2.0}} = 0.06, \beta = 0.44, \gamma = 0.01$. While for AMD, these are: $\alpha_{f_{2.0}} = 0.06, \beta = 0.0, \gamma = 0.01$. Tables 6 and 7 show the relative error using 99% confidence interval. The results indicate the high accuracy of our proposed model of Equation (22) and helps to reduce the relative error in both Intel and AMD processors, which is usually less than 5% and always under 9%. Finally, Figure 9 presents a comparison between the power consumption obtained through the power meter (Act) and the power consumption computed using Equation (22) (Est). We can notice that both have identical consumption with a difference of 3W in worse cases.

6. CONCLUSION AND PERSPECTIVES

Traditionally, the power consumption of multi-core processors was assumed to be the sum of power drain of their consistent cores. In this paper, we evaluated this assumption through a set of CPU-bound and I/O bound observations. We showed that, in most cases, the above mentioned assumption overestimates the power consumption. Furthermore, we discovered that certain components of a multi-core processor has a constant power consumption behavior regardless of number of involved cores. Consequently, we went one step further and proposed a component-based power consumption model for multi-core processors. Moreover, un-

Processor	F[GHz]	Voltage[V]	%Reduction (r)		
			2 cores	3 cores	4 cores
Intel SpeedStep	2.0	1.104	0.06	0.07	0.08
Intel SpeedStep	2.5	1.104	0.30	0.30	0.30
AMD Cool'n'Quiet	2.5	1.05	0.06	0.07	0.08

Table 4: Power reduction factor r of Intel SpeedStep and AMD Cool'n'Quiet for quad-core processors.

CPU type	Frequency (GHz)	RAM	Hard Disk
Intel Core 2 Duo E6400	2.13	DDR_2, 8GB	250GB
Intel Xeon L5420 Quad core	2.0 (min) – 2.5 (max)	DDR_2, 16 GB	320GB
AMD Phenom II X4 955 Quad core	2.0 (min) – 3.0 (max)	DDR_3, 8GB	160 GB

Table 5: Processor's characteristics

like the existing models, we studied energy-efficient mechanisms and proposed an empirical power reduction model both for AMD and Intel processors.

Although we believe that the proposed component-based models are suitable to a wide range of processor, as a future work, it could be interesting to extend this work to the case of processors consisting of more than 4 cores and investigate further their corresponding energy-efficient mechanisms.

7. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's 7^{th} Framework Programme in the context of the FIT4Green project.

8. REFERENCES

[1] `http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx`.

[2] `http://www.zes.com/english/products/one-to-eight-channel-precision-power-analyzer-lmg500.html`.

[3] `http://ark.intel.com/Product.aspx?id=33929`.

[4] `http://www.devin.com/lookbusy/`.

[5] L. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33 –37, 2007.

[6] R. Berrendorf and B. Mohr. PCL - The Performance Counter Library Version 2.2, Jan. 2003.

[7] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade. Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of 24th ACM Int'l Conf. on Supercomputing*, ICS '10, pages 147–158. ACM, 2010.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Int'l Symp. on Computer Architecture*, pages 83 – 94, 2000.

[9] A. Chandrakasan and R. Brodersen. Minimizing power consumption in digital CMOS circuits. *Proceedings of the IEEE*, 83(4):498 –523, Apr. 1995.

[10] A. P. Chandrakasan and R. W. Brodersen. Minimizing power consumption in cmos circuits. Technical report, University of California at Berkeley, 1995.

[11] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual Int'l Symposium on Computer Architecture*, pages 13–23. ACM, 2007.

[12] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of Int'l Symp. on Low Power Electronics and Design*, pages 38 –43. ACM/ IEEE, 2007.

[13] C. Hewlett-Packard, C. Intel, C. Microsoft, L. Phoenix Technologies, and C. Toshiba. Advanced configuration and power interface specification, 2010.

[14] C.-T. Hsieh, Q. Wu, C.-S. Ding, and M. Pedram. Statistical sampling and regression analysis for RT-Level power evaluation. In *Proceedings of Int'l Conf. on Computer-Aided Design*, pages 583 –588.

[15] C.-H. Hsu, J. J. Chen, and S.-L. Tsao. Evaluation and modeling of power consumption of a heterogeneous dual-core processor. In *Proceedings of Int'l Conf. on Parallel and Distributed Systems*, pages 1–8, 2007.

[16] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski. The design and implementation of PowerMill. In *Proceedings of the Int'l Symp. on Low Power Design*, pages 105–110. ACM, 1995.

[17] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors, 2001.

[18] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. Energy management for commercial servers. *Computer*, 36(12):39 – 48, 2003.

[19] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: eliminating server idle power. In *Proceeding of the 14th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 205–216. ACM, 2009.

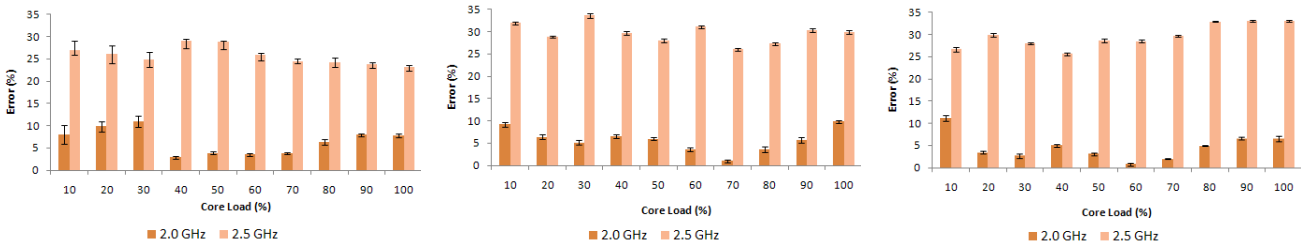
[20] V. Pallipadi. Enhanced Intel SpeedStep Technology and Demand-Based Switching on Linux, Feb 2009.

[21] G. Qu, N. Kawabe, K. Usarni, and M. Potkonjak. Function-level power estimation methodology for microprocessors. In *Proceedings of Design Automation Conference*, pages 810–813, 2000.

[22] J. Russell and M. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors. In *Proceedings of Int'l Conf. on Computer Design*, pages 328 –333, 1998.

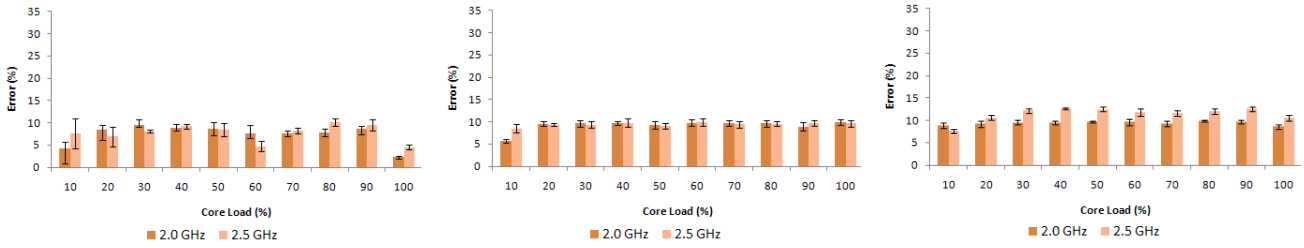
[23] K. Singh, M. Bhadauria, and S. A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37:46–55, July 2009.

[24] P. E. West. Core monitors: Monitoring. Master's thesis, THE FLORIDA STATE UNIVERSITY, 2008.



(a) 2 cores (b) 3 cores (c) 4 cores

Figure 10: Relative error percentage of Equation (2) using 99% confidence interval for Intel quad-core.



(a) 2 cores (b) 3 cores (c) 4 cores

Figure 11: Relative error percentage of Equation (2) using 99% confidence interval for AMD quad-core.

4*Load (%)	Core 2 Duo		Quad core											
	Two cores		Two cores				Three cores				Four cores			
	f=2.13		f=2.0		f=2.5		f=2.0		f=2.5		f=2.0		f=2.5	
	mean	+/- ci	mean	+/- ci	mean	+/- ci	mean	+/- ci	mean	+/- ci	mean	+/- ci	mean	+/- ci
10	1.56	3.51	3.58	0.61	7.24	0.46	5.12	0.37	3.17	4.34	0.19	2.06	-1.25	0.63
20	-0.58	1.48	5.47	0.41	-7.16	1.01	1.24	0.22	2.19	1.44	-2.05	1.47	2.39	0.4
30	3.76	4.63	6.72	0.49	-7.39	1.06	0.11	1.02	5.86	3.37	-4.75	1.74	0.37	0.58
40	3.40	1.76	-0.97	0.13	5.9	0.49	2.01	0.75	1.72	3.24	-0.34	3.38	-3.00	0.29
50	4.05	1.41	0.42	0.58	5.01	0.29	1.56	0.52	0.86	1.2	-4.01	0.93	0.50	0.61
60	1.95	1.72	0.03	0.71	1.4	0.52	-1.43	0.56	4.53	2.96	-3.09	3.72	0.58	0.37
70	0.38	1.38	0.01	0.98	-0.4	1.06	-4.11	0.26	-0.83	0.88	-4.00	1.20	2.76	0.57
80	-0.17	0.82	2.86	0.74	-0.83	0.68	-1.16	0.34	0.78	1.40	-2.34	3.11	6.98	0.22
90	-2.00	2.11	4.47	0.67	-1.7	0.52	1.29	0.52	3.71	4.52	-0.45	2.46	7.57	0.54
100	0.37	1.90	4.39	0.79	-2.28	0.61	5.47	0.52	3.45	1.29	-2.09	1.83	7.68	0.47

Table 6: Intel Processors: Relative error percentage of Equation (22) with 99% confidence interval (ci).

4*Load (%)	Two cores										Three cores										Four cores									
	f=2.0		f=2.3		f=2.5		f=2.7		f=3.0		f=2.0		f=2.3		f=2.5		f=2.7		f=3.0		f=2.0		f=2.3		f=2.5		f=2.7		f=3.0	
	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci	mean	ci
10	1.17	5.63	-2.75	3.21	2.68	4.04	-4.25	3.77	-4.48	1.99	3.28	2.78	2.70	2.77	2.59	3.39	3.03	2.05	3.10	4.29	-2.92	2.65	2.59	5.04	-2.16	5.12	4.08	1.39	-0.85	5.12
20	1.75	2.24	0.36	3.22	1.98	2.22	3.79	1.96	2.00	1.00	3.05	2.08	2.04	1.47	0.22	2.18	4.75	1.65	2.16	2.60	0.33	3.18	5.56	0.30	-2.98	3.16	2.76	0.75	-3.85	2.69
30	0.07	3.73	-2.28	3.15	0.41	1.98	2.98	1.69	-5.78	2.22	4.24	1.32	3.06	1.90	-3.39	2.00	5.72	1.64	6.09	0.50	-2.57	1.93	-0.17	1.79	-4.81	0.80	-1.57	0.77	-3.79	0.61
40	3.15	4.27	-0.26	4.27	3.08	1.67	3.13	1.89	-3.20	0.56	6.61	0.85	1.95	1.43	-1.72	2.41	6.14	0.75	0.82	2.21	-2.66	0.98	-2.08	0.87	-4.56	0.66	-5.49	0.37	-5.87	0.23
50	3.02	4.48	-1.86	1.88	3.37	2.01	2.73	1.55	-0.60	1.33	4.26	1.66	2.68	1.15	-3.16	1.14	5.77	0.64	-2.37	1.00	-4.42	0.78	-5.13	0.81	-4.89	0.37	-7.03	0.22	-5.95	0.19
60	0.78	1.87	-3.15	3.90	-1.44	1.83	3.92	2.51	-3.80	1.06	4.47	1.09	2.56	1.10	-3.80	1.46	5.72	0.54	-3.77	0.60	-4.52	0.58	-4.86	0.79	-3.88	0.48	-7.15	0.51	-5.23	0.08
70	1.08	1.60	-3.55	1.82	2.16	2.10	2.60	1.65	-3.67	1.02	6.23	1.26	3.46	0.82	-3.06	1.21	5.96	0.54	-3.29	1.25	-3.80	1.35	-5.81	0.64	-2.63	0.54	-7.59	0.14	-3.67	0.34
80	1.47	1.41	-1.67	2.08	2.51	1.55	2.18	2.37	-1.51	1.07	5.13	1.12	3.92	0.72	-1.92	1.16	6.09	0.56	-4.41	0.48	-2.82	1.25	-6.18	0.35	-5.64	0.36	-7.28	0.21	-2.92	0.33
90	1.92	1.90	-0.74	1.49	3.54	1.42	4.52	1.27	-1.63	0.74	4.25	0.96	4.43	1.31	-1.82	1.22	5.07	0.41	-5.65	0.68	-2.07	0.68	-5.46	0.38	-1.79	0.28	-7.07	0.13	-0.57	0.22
100	-1.72	4.24	-3.34	1.05	-1.48	1.31	1.34	1.90	-3.53	1.03	2.50	1.00	2.46	1.44	-3.29	1.43	3.90	0.36	-6.20	0.53	-2.16	0.36	-5.92	0.31	-1.74	0.44	-7.40	0.27	-0.41	0.42

Table 7: AMD Processors: Relative error percentage of Equation (22) with 99% confidence interval (+/- ci).