J Supercomput (2014) 70:1451–1476
DOI 10.1007/s11227-014-1236-4

Energy measurement, modeling, and prediction for processors with frequency scaling

Thomas Rauber · Gudula Rünger · Michael Schwind · Haibin Xu · Simon Melzner

Published online: 17 June 2014

Abstract The energy consumption is an important aspect of today's processors and a large variety of research approaches deal with reducing the energy consumption for specific application codes on different platforms under certain constraints. These research approaches are based on energy information acquired by very different means, such as hardware settings with power-meters, software methods with hardware counters available for more recent CPUs, or simulations based on theoretical models. In this article, all of these energy acquisition methods are investigated and compared. As application programs, we consider the SPEC CPU2006 integer and floating-point benchmark collections, which represent a large variety of applications from different areas. The investigations are done for single multicore CPUs with the goal to get more insight into their energy consumption behavior. An experimental evaluation is performed on three recent processor types with dynamic voltage–frequency scaling. The article compares the measured energy and the energy provided by hardware counters with the energy predicted by simulation models. The comparison shows that the simulation models are able to capture the energy consumption quite accurately.

Keywords Dynamic voltage–frequency scaling · DVFS · SPEC CPU2006 benchmarks · Energy measurement · Energy models

T. Rauber (✉) · S. Melzner
Computer Science Department, University Bayreuth, Bayreuth, Germany
e-mail: rauber@uni-bayreuth.de

G. Rünger · M. Schwind · H. Xu
Computer Science Department, Chemnitz University of Technology, Chemnitz, Germany

1 Introduction

Energy-aware computing and the efficient use of compute resources are now accepted to be as important for application codes as performance aspects. Energy efficiency has already been a critical concern in digital circuit design for the last two decades. This has led to several power-aware system features, including multicore-on-a-chip processors, core-independent functional units, dynamic frequency and voltage scaling, or clock gating, which are mainly aimed at a reduction of the energy consumption of the processors. To fully exploit the hardware capabilities for decreasing the energy consumption of application software running on that hardware, information about the energy consumption has to be available. Reliable energy models and measures are needed to plan and evaluate the software energy behavior on specific hardware.

One of the most effective hardware energy minimization techniques is dynamic voltage and frequency scaling involving a dynamic adjustment of the clock frequency and the corresponding supply voltage. Dynamic voltage–frequency scaling (DVFS) reduces the dynamic power consumption which is that part of the power needed to make a processor work. The static power is the power of the processor when it is not active and mainly includes the leakage power. There is also a considerable amount of power consumed by the entire hardware system and all its devices. The energy consumption of application code can be reduced by either decreasing the execution time, e.g., by algorithmic methods, which may result in a smaller energy consumption, or by scaling the frequency to decrease the power consumption, which may however increase the execution time. Since a low execution time is still an important goal, for instance when simulating large applications or when a predefined response time is required, energy-awareness and performance awareness have to be treated together. Thus, in addition to performance data, the availability of reliable energy data is a prerequisite to tune application code towards energy efficiency.

Measuring, evaluating or modeling energy consumption for either single specific applications on selected hardware platforms or an entire range of applications emphasizing more on the hardware facility is an active research area. The goal is to acquire information about the energy consumption with the aim to understand, to reduce or even to minimize the energy consumption under specific constraints. This requires a very solid foundation of energy data which are reliable in the sense that they reflect the reality qualitatively and quantitatively. Thus, the challenge is how to get good-quality energy data. The methods used for gathering energy information of application codes running on modern hardware platforms are quite different and three main categories have evolved, which are hardware measurements with specific power-meters, software measurements with hardware counters available in more recent processors, and simulations based on theoretical energy models.

In this article, we consider all the three methods for gathering energy data with the goal to investigate the correspondence or difference of the energy data provided by the methods. A contribution of the article is a detailed comparison of the energy measurement using power-meters and using the running average power limit (RAPL) interface of recent Intel processors. As application we have chosen the SPEC CPU2006 benchmarks, which are well-known to the community and reflect properties of typical application codes. The measurements with both of measurement techniques show

a good correspondence, which allows us to use the energy information gathered by the faster and easier-to-use hardware counters. In contrast to earlier work, such as presented in [1,2], the comparison focuses on the frequency scaling feature of the processors and is performed for a large set of complete application programs (the SPEC CPU2006 integer and floating-point programs). The correlation of the data acquired by these two measurement methods is crucial for getting experimental data on which a power prediction method can be based. A second contribution of the article is the comparison of the measured energy values with the energy values provided by different DVFS energy models. In particular, a physical energy model and a heuristic energy model are investigated. These energy models can be used for an a priori energy estimation and as a basis for simulating the energy consumption of application programs.

The rest of the article is structured as follows. Section 2 describes measurement techniques for the energy consumption of processors and compares these techniques and their resulting energy values for the SPEC CPU2006 benchmarks. Section 3 presents energy measurements for the SPEC benchmarks on different modern processors. Section 4 describes different models for capturing the energy consumption of DVFS processors. Section 5 investigates how well these energy models are suited for predicting the energy consumption of processors. Section 6 discusses related work. Section 7 concludes the paper.

2 Measurement techniques for the energy consumption of DVFS processors

In this section, we describe the techniques with which we have measured the power and energy consumption of DVFS processors and present the resulting energy data. Hardware measurement techniques using specialized power-meters are described in Sect. 2.1. A software-based measurement technique based on accessing the hardware counters of the processors is described in Sect. 2.2. A comparison of the measured energy data is done in Sect. 2.3.

Modern microprocessors, such as the Intel Core i7 processors, incorporate a power management technology which supports different power management states: performance states (P-states), throttle states (T-states), idle states (C-states) and sleep states (S-states) [3]. P-states are predefined sets of paired frequency and voltage combinations at which an active core can operate [4]; the various P-states supported are implemented using a combination of dynamic frequency scaling (DFS) and dynamic voltage scaling (DVS). A C-state is an idle state in which parts of the processor are powered down to save energy. Various C-states are supported by Intel processors, and a higher numbered C-state indicates more power savings.

For the experiments, three different Intel Core i7 processors have been used:

(i) an Intel Core i7-2600 processor with the Sandy Bridge architecture;
(ii) an Intel Xeon CPU E3-1225 V2 processor with the Ivy Bridge architecture;
(iii) an Intel Core i7-4770 with the Haswell architecture. Table 1 describes some details of the processor architectures.

Table 1 Characteristics of the processors used for the experimental evaluation

	Core i7-2600	Xeon E3-1225V2	Core i7 4770
Architecture	Sandy Bridge	Ivy Bridge	Haswell
Minimum frequency (GHz)	1.6	1.6	0.8
Maximum frequency (GHz)	3.2	3.2	3.4
TDP (W)	95	77	84
Step size frequency (MHz)	100	100	200
Physical cores	4	4	4
Hyperthreading	Yes	No	Yes
Virtual cores	8	4	8
L1 data cache (KB)	32	32	32
L2 cache (KB)	256	256	256
L3 shared cache (MB)	8	8	8
RAM size (GB)	8	8	8

2.1 Power measurement with power-meter

A precise method to capture the power consumption of computer systems is the direct measurement using a power-meter. We have implemented this approach using the data acquisition system National Instruments NI9205, which has been integrated into an NI CDAQ9181 chassis to enable the transfer of the measured power values via Ethernet to a separate system for data analysis, see Fig. 1 for an illustration of the overall configuration and Fig. 2 for a photo of the experimental setup. The power measurement technique is similar to the technique used by the PowerPack framework [5].

The NI9205 enables a fine-grain power measurement of different components of a computer system. In particular, it is possible to isolate the power consumption of the CPU, the main memory, disks, fans, etc. by accessing the corresponding connectors used for the supply voltage of an individual component and measuring the power consumption at this connector. For example, the CPU is powered through a +12VDC pin whereas the memory is powered through a +3.3VDC pin. The overall power

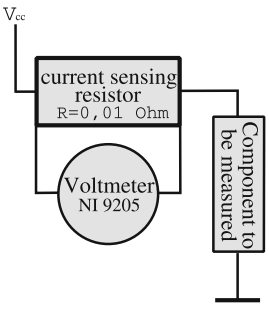

Fig. 1 Illustration of the power measurement using the NI9205 device for measuring the supply voltage Vcc for the different wires

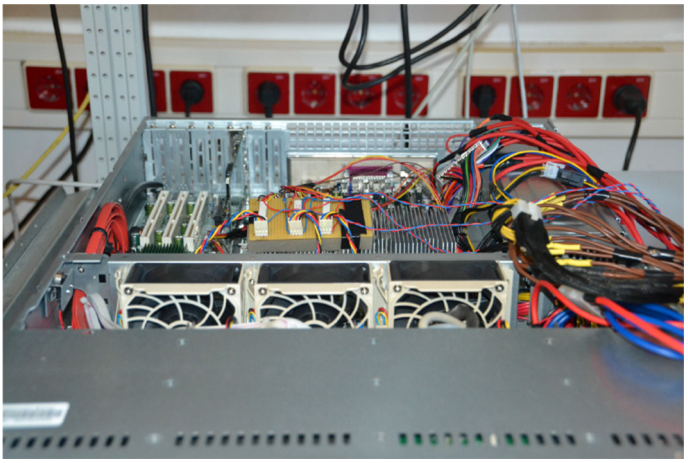

Fig. 2 Photo of the power measurement experimental setup using the NI9205 device

consumption of the computer system is the sum of the power consumption of the individual components.

For the actual power acquisition and profiling, the software tool LabView [6] has been used for which we have configured several modules operating in a client–server fashion so that the collection of the power data is automated. The client program initiates the LabView server program to acquire power data and then starts the application program for which the power profile is requested. After the termination of the application program, the LabView module saves the collected power consumption data to disk and initiates the post-processing of the data. LabView modules were also needed for the synchronization of the power profiling process with the running application program. The software modules developed are also able to capture the power consumption of specific parts of the application program to be analyzed and to determine which part of a computer system has which power consumption at which point of program execution. As example, Fig. 3 shows the power consumption of the SPEC CPU2006 benchmark libquantum running on a Core i7 Ivy Bridge architecture measured with the NI9205 power-meter. The time interval between 50 and 50.5 s is shown, which has

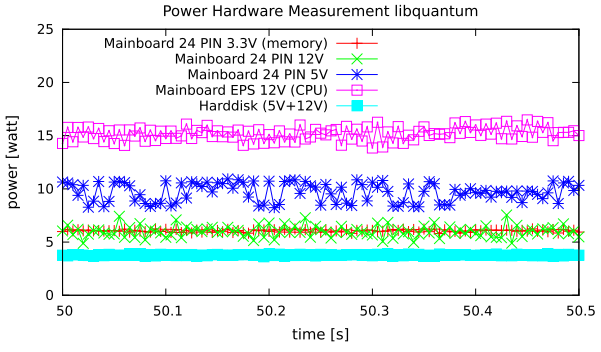

Fig. 3 Detailed results of the power measurement for a specific time interval of the SPEC benchmark libquantum using the NI9205 device on an Intel Ivy Bridge processor

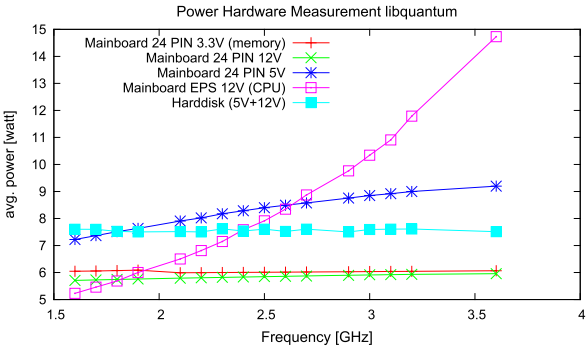

Fig. 4 Detailed power measurement for the SPEC benchmark libquantum executed at different frequencies; measurement using the NI9205 device on an Intel Ivy Bridge processor

been selected arbitrarily. Other intervals show a similar behavior. The total runtime of the benchmark is 292 s in the turbo mode at 3.6 GHz. The following five connectors have been measured: (1) the 24 pin ATX 3.3V main-board connector supplying the memory modules; (2) the 24 pin ATX 12V main-board connector supplying peripheral devices and the CPU; (3) the 24 pin ATX 5V main-board connector also supplying peripheral devices and the CPU; (4) the main-board 12V EPS connector which is mainly used by the CPU; (5) the 12V and 5V supply lines for the hard disk. Figure 3 shows that the power consumption of the memory modules is quite small compared to the power consumption of the CPU, and the power consumption due to disk accesses is even smaller.

Figure 4 shows the power consumption for the different connectors for the SPEC CPU2006 benchmark libquantum depending on the frequencies. Especially the power consumption of the 12V EPS connector, which is the main power supply of the CPU, is significantly increasing with the frequency. The power consumption of the ATX 5V main-board connector, which is also partly used by the CPU, also increases slightly. The power consumption of the remaining connectors remains nearly constant over the range of available frequencies, since they are not affected by the frequency change.

The power-meter approach can be extended to capture the power consumption of parallel systems, such as clusters, by measuring the power consumption of the nodes of the cluster system separately and adding up the measured power consumption of the nodes. Figure 5 shows the runtimes and energy consumption of selected SPEC CPU2006 benchmarks. However, this would require a significant amount of measuring devices, especially for larger parallel systems. An alternative approach is to use software techniques with hardware counters as described in the Sect. 2.2 requiring much less overhead. The question arises how accurate the measurement with hardware counters is in comparison with a measurement with a power-meter. We address this question in Sect. 2.3 (Fig. 5).

2.2 Power measurement with RAPL sensors

Many recent processors provide hardware counters which capture the energy consumption. Starting with the Sandy Bridge architecture, Intel has introduced the RAPL

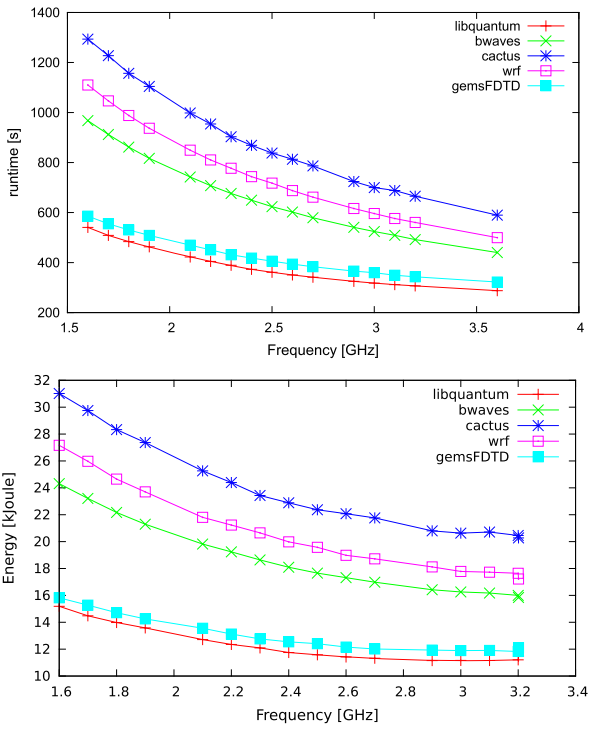

Fig. 5 Runtime and energy consumption of selected SPEC CPU2006 benchmarks with varying frequencies. For the energy, the total energy consumption is shown, collected with the NI9205 device on an Intel Ivy Bridge processor

feature, which provides sensors that allow the measurement of the power consumption of CPU components [2,3]. The Core i7 processor family has two power planes on chip, which are PP0 containing the processor cores and all caches, and PP1, also referred to as Uncore, containing additional devices, such as graphics devices or the Package Control Unit (PCU) [2]. The corresponding RAPL sensors are RAPL_PPO and RAPL_PP1 measuring the power consumption of the processor core and the processor uncore. In addition, there are the RAPL sensors RAPL_DRAM and RAPL_PKG for measuring the power consumption of the memory controller and the whole CPU package, respectively.

RAPL sensors can be accessed by control registers, known as Model-Specific Registers (MSRs), which are updated in intervals of about 1 ms [3]. The MSR can be accessed by a pair of instructions *rdmsr* and *wrmsr*. These instructions are privileged (privilege level 0) and are to be executed in kernel mode. The MSRs provide information about the energy status of the PP0 and PP1 power planes using the specific registers MSR_PP0_ENERGY_STATUS and MSR_PP1_ENERGY_STATUS. The MSR MSR_PKG_ENERGY_STATUS captures the energy status of the whole CPU package. The corresponding energy status unit is obtained via the MSR_RAPL_POWER_UNIT MSR; the default value is 15.3 μJ.

Our investigations include power and energy measurements exploiting RAPL sensors for the SPEC CPU2006 benchmarks. For the RAPL-based experiments, we have used the likwid toolset (Version 3.0) [7], which provides access to the MSRs introduced

above. In the following subsection, the measurement methods with power-meters and RAPL sensors are compared and evaluated. The goal is to clarify the influence of the measurement overhead on the measurement results.

2.3 Comparison of the measurement techniques

Hardware counters, such as the MSR registers, can capture only the power consumption of the CPU. On the other hand, measurements with specialized devices, such as the NI9205 system, enable a more detailed measurement that can also take the memory system, fans, and peripheral devices into account. However, a significant hardware and software overhead is required to perform accurate power-meter measurements with such a device. One might argue that the main contribution to power consumption comes from the CPU and that the measurement with hardware counters is good enough to capture the relevant effects. In this context, it is interesting to investigate how accurate the power measurement with hardware counters is compared with detailed measurements using special power-meter devices. We address this issue in this section and present a comparison for selected application programs.

To set the frequencies of the cores to a fixed value, we have used the *cpufreq_set* tool; the minimum and the maximum core frequencies have been set to the same value. During program execution, the frequency of each core can be observed with the *i7z* tool. The runtime experiments have been performed with no other application program running on the machine, i.e., interferences can only come from jobs of the operating system. To reduce the effect of such interferences, the runtime experiments have been performed several times.

For the Sandy Bridge architecture, the operational frequency can be set between 1.6 and 3.4 GHz in 100-MHz steps. The same holds for the Ivy Bridge architecture with the exception that the maximal frequency is 3.2 GHz. For the Haswell architecture, the frequency can be set between 0.8 and 3.4 GHz; the stepsize is usually 200 MHz with two exceptions (1.4/1.5 GHz and 2.7/2.8 GHz).

Figure 6 compares the two energy measurement techniques described above. The measurement has been performed for the SPEC benchmark libquantum on an Intel Core i7 Sandy Bridge processor. The benchmark runs have been repeated 20 times with the "test" input. The power measurement shown in Fig. 6 using the NI 9205 acquisition system takes only the main-board 12V EPS connector into consideration, which is the main power supply of the CPU. The 12V ATX and the 5V ATX connector, which partially supply the CPU and also other periperal devices, are not taken into consideration, as it cannot be determined which part of the power actually goes to the CPU. On the other hand, the measurement with the RAPL interface captures the complete power consumption of the CPU. The diagram shows a good match of the energy values obtained with both techniques for some of the frequencies. For other frequencies, the energy values obtained with RAPL are up to about 20 % larger than the energy values obtained with the NI 9205 device. A reason for these deviations could be that there is an additional power flow to the CPU via the 12V and the 5V ATX connectors, which is not captured. Our investigations have shown that these deviations are smaller for most of the other SPEC benchmarks on the Sandy Bridge

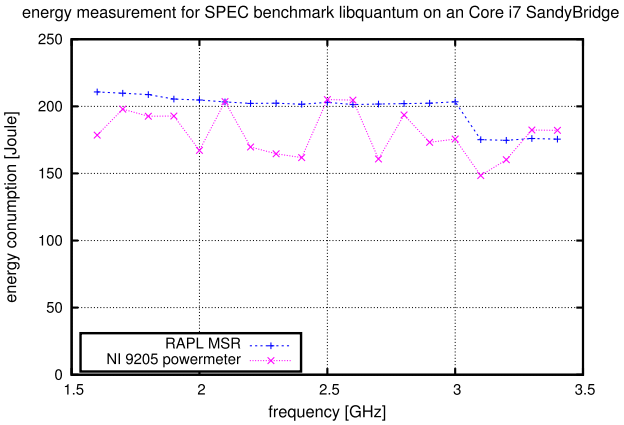

Fig. 6 Comparison of the results of the energy measurement with power-meters and the energy measurement using RAPL for the SPEC benchmark libquantum on an Intel Core i7 Sandy Bridge processor

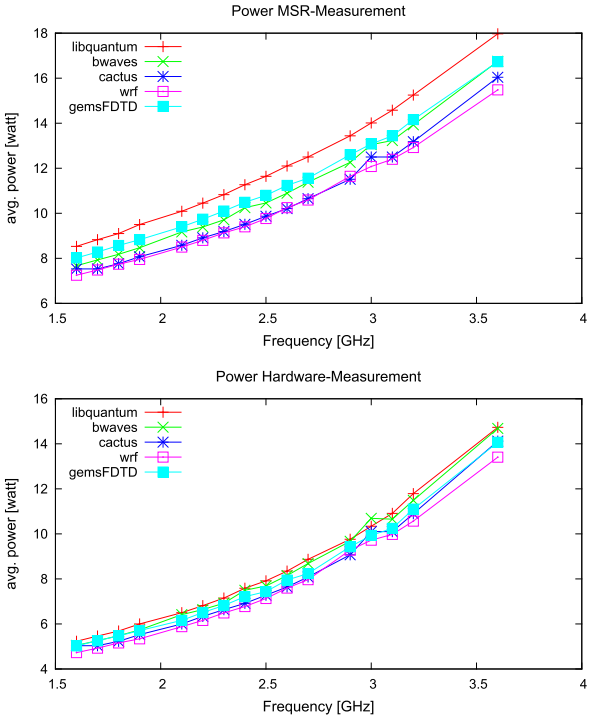

Fig. 7 Comparison of power measurement with MSR registers (*top*) and NI9205 device (*bottom*) on an Intel Ivy Bridge processor

architecture. The differences in the deviations for different frequencies are not present for the Ivy Bridge processor as described in the following.

Figure 7 shows the power consumption obtained with the two measurement methods for the Core i7 Ivy Bridge architecture using five selected SPEC benchmarks. The figure shows that the power consumption increases more than linearly when the

frequency is increased. Both measurement techniques can capture this behavior. The figure also shows that the two measurement methods lead to similar values for the power consumption. In particular, the relative order of the power consumption of the different benchmarks is the same for both measurement techniques. However, the power values obtained with RAPL are systematically larger than the power values obtained with the NI 9205 device for the same reason as mentioned for Fig. 6. In particular, the power measured with the RAPL interface is typically about 3 W larger than the power measured with the NI 9205 device using only the 12V EPS connector. Adding the power consumption of the 12V ATX and 5V ATX connectors to the power measured with the NI 9205 device would lead to larger values as measured with the RAPL interface. In summary, it can be concluded that the power measurement via RAPL is accurate enough to replace the usage of power-meters.

3 Energy consumption of the SPEC CPU benchmark programs

In this section, we present results of power and energy measurements for the entire set of applications in the the SPEC CPU2006 benchmark suite, see Figs. 8, 9, and 10. Based on the findings of the previous section, the RAPL interface has been used to obtain the power and energy values. The likwid toolset [7] has been used to access the MSR registers of the different architectures.

The SPEC CPU2006 benchmark suite has been developed with the goal to capture the performance of desktop systems and single-processor servers. The benchmark suite consists of integer and floating-point benchmarks, which are real programs covering different application areas for computer systems. The benchmarks are sequential C, C++, or Fortran programs. The integer programs include, for example, a compression program (bzip2), a C compiler (gcc), a video compression program, a chess game, and an XML parser. The floating-point programs include, for example, several simulation programs from physics, a speech recognition program, a ray-tracing program (povray), as well as programs from numerical analysis and a linear programming algorithm (soplex), see, e.g., [8] and http://www.spec.org for more details. In the experimental evaluation, the benchmarks have been compiled with gcc 4.7.2 using the compiler option -ftree-parallelize-loops=4, enabling an automatic parallelization at loop level.

Figure 8 shows the runtimes (top) in seconds, the energy consumption (middle) in Joule, and the power consumption (bottom) in Watt as functions of the frequency for the SPEC CPU2006 integer benchmarks on an Intel Core i7 Sandy Bridge architecture. All available frequencies are shown. The figure shows that the runtime decreases nearly linearly with increasing frequencies for all benchmarks. Only for smaller frequencies, the increase of the execution time with a reduced frequency is more than linear. The diagram shows that for most of the SPEC CPU2006 integer benchmarks, there is only a slight variation of the energy consumption for the entire range of the frequencies possible. The power consumption values P have been calculated from the energy values E obtained from the MSR registers and the runtime t measured also with hardware registers using $E[\mathrm{J}] = P \times t[\mathrm{W \cdot s}]$. Looking at the power consumption, it can be observed that different applications lead to slightly different power consumptions, i.e., there is a small dependence of the power consumption from the characteristics

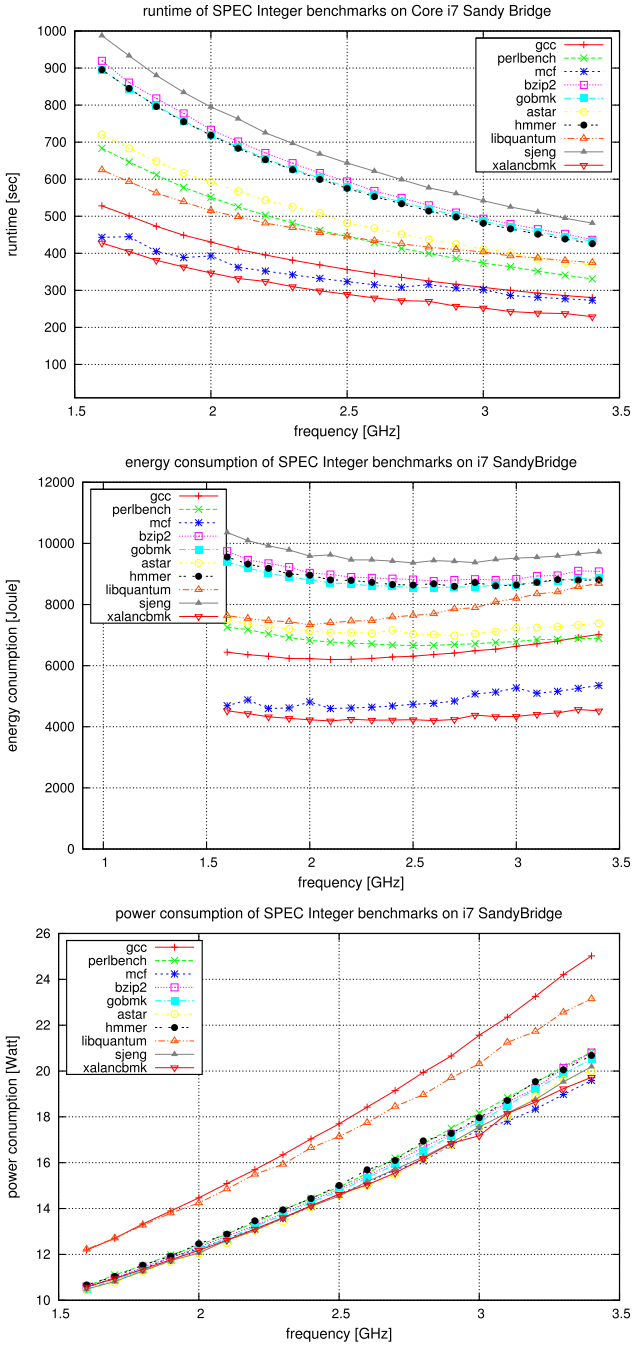

Fig. 8 SPEC CPU2006 integer benchmarks on an Intel Core i7 Sandy Bridge processor: runtime (*top*), energy consumption (*middle*), and power consumption (*bottom*) for varying frequencies

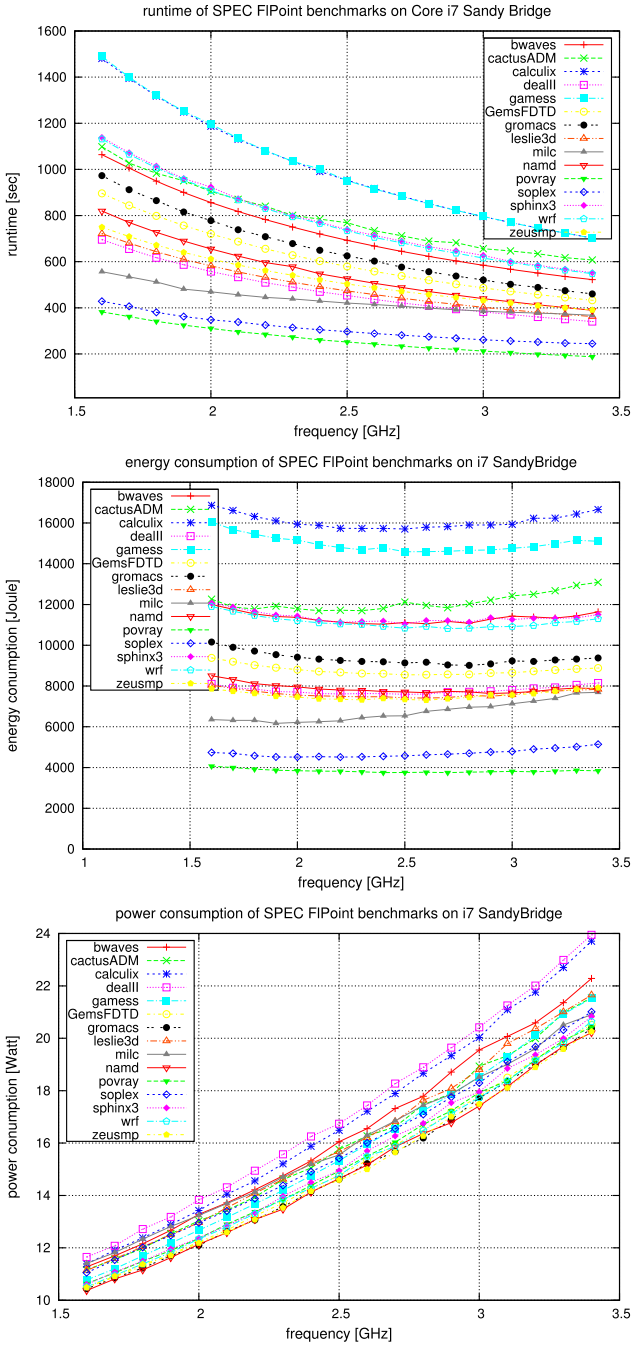

Fig. 9 SPEC CPU2006 floating-point benchmarks on an Intel Core i7 Sandy Bridge processor: runtime (*top*), energy consumption (*middle*), and power consumption (*bottom*) for varying frequencies

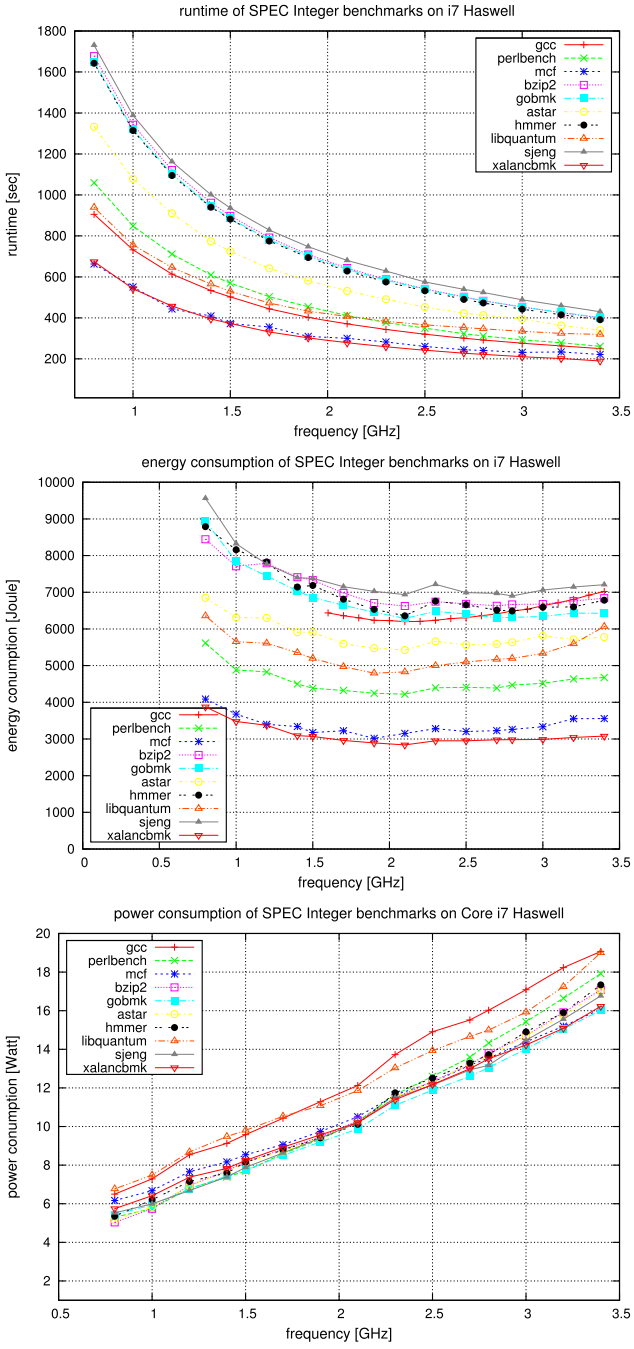

Fig. 10 SPEC CPU2006 integer benchmarks on an Intel Core i7 Haswell processor: runtime (*top*), energy consumption (*middle*), and power consumption (*bottom*) for varying frequencies

of the application. Two benchmarks (gcc and libquantum) lead to a larger power consumption than the rest of the integer benchmarks. This shows that in the case of the SPEC CPU2006 integer benchmarks, the power consumption mainly depends on characteristics of the hardware and that the different energy consumptions for the different benchmarks are mainly caused by the different execution times.

Figure 9 shows the runtimes (top), energy consumption (middle), and power consumption (bottom) for the SPEC CPU2006 floating-point benchmarks for different frequencies on an Intel Core i7 Sandy Bridge architecture. The SPEC CPU2006 floating-point benchmarks typically have a slightly larger execution time than the integer benchmarks. Accordingly, a larger energy-consumption results. The energy consumption shows a similar behavior as for the integer benchmarks and there is no great variation of the energy consumption in dependence of the frequency. Compared to the integer benchmarks, a slightly larger variation of the power consumption can be observed for the different SPEC floating-point benchmarks. This suggests that there is a larger dependence of the power consumption from the usage of processor resources by the specific application.

Figures 10 and 11 show the runtimes (top), energy consumption (middle), and power consumption (bottom) of the SPEC CPU2006 integer and floating-point benchmarks for different frequencies on an Intel Core i7 Haswell architecture. Similar to the Sandy Bridge architecture, the figures show a decrease of the runtime of the different benchmarks which is nearly linear to the operational frequency. For smaller frequencies, the more-than-linear increase of the execution time with reduced frequency is stronger than on the Sandy Bridge architecture, since the Haswell architecture allows smaller frequencies. Especially for larger frequencies, most of the benchmarks are faster on the Haswell architecture than on the Sandy Bridge architecture, presumably due to the more recent architecture with a larger L3 cache. Especially for small frequency values, the energy consumption increases more visibly than for the Sandy Bridge architecture, caused by the increase in execution time. The power consumption on the Haswell architecture is typically smaller than on the Sandy Bridge architecture for the same frequency. The same holds for the energy consumption, i.e., the Haswell architecture usually needs less energy for the same benchmarks than the Sandy Bridge architecture using the same frequency.

For all SPEC CPU2006 benchmarks and both architectures, the diagrams of the runtime, the power and the energy consumption show a similar qualitative behavior as a function of the frequency. The runtime is a decreasing, slightly convex function with the highest runtime for the smallest available frequency and the smallest runtime for the highest available frequency. The power consumption is a slightly convex, increasing function depending on the frequency. As a result, the energy consumption, being the power integrated over the runtime, is a convex function that is slightly descending for smaller frequencies and slightly ascending for higher frequencies. This effect is more distinct for the Haswell processor. These energy functions have a minimum in the frequency range between 2 and 2.7 GHz, varying for the different benchmarks and architectures. Since the SPEC CPU2006 benchmarks cover a wide range of different applications, one may conclude that it is a typical behavior to have an energy minimum and to achieve the minimum, a suitable frequency should be used. However, the minimum of the runtime, being at the largest frequency, and the minimum

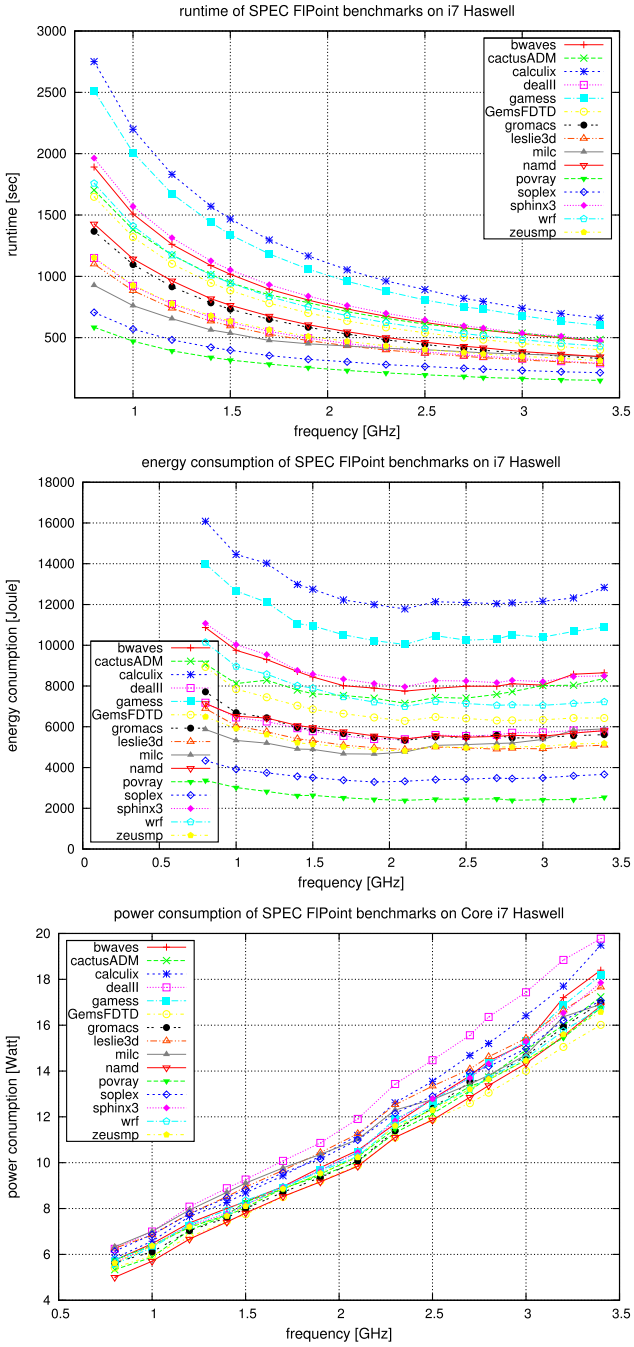

Fig. 11 SPEC CPU2006 floating-point benchmarks on an Intel Core i7 Haswell processor: runtime (*top*), energy consumption (*middle*), and power consumption (*bottom*) for varying frequencies

of the energy consumption do not coincide. Thus, the user has to decide whether to minimize the runtime or the energy consumption before selecting a frequency.

4 Energy models with frequency scaling

Many different energy models have been proposed and used for simulating the energy consumption of application programs on CPUs, see [9,10] for an overview. These models usually concentrate on the dynamic power consumption, which used to be the most dominant part in earlier processors, and more recently also include the static power consumption, memory power consumption or other significant power aspects of a specific CPU or computer system. The dynamic power consumption is related to the supply voltage and the switching activity during the computing activity of the processor. The static power consumption captures the leakage power consumption as well as the power consumption of peripheral devices. The total power consumption includes both power components.

For DVFS processors, the power consumption depends on the operational frequency f, see, e.g., Fig. 8 for an experimental evaluation of this dependence. In this section, we investigate two models simulating the energy consumption, a physically based model which has been proposed in the literature and a heuristic model that is derived from the experimental results in Sect. 3.

4.1 Physical energy model with frequency scaling

Analytical energy models describe the power consumption of a processor by power models developed for digital circuits used for the construction of the processors. The energy consumption is calculated by a multiplication with the runtime of an application program. This class of energy models is typically derived from physical considerations of the expected power behavior, and the models are used within optimization or scheduling methods which are aimed at the minimization of the energy consumption. In the following, we exploit a physical energy model that has been proposed in the literature, see [9,10]. In this article, we use this model and determine parameters for the SPEC benchmarks.

The specific energy model considered approximates the dynamic power consumption by $P_{\text{dyn}} = \alpha \cdot C_{\text{L}} \cdot V^2 \cdot f$, where α is the switching probability, C_{L} is the load capacitance, V is the supply voltage, and f is the operational frequency. The static power consumption is intended to capture the leakage power consumption which consists of several components, including sub-threshold leakage, reverse-biased junction leakage, gate-induced drain leakage, gate-oxide leakage, gate-current leakage, and punch-through leakage [10]. The exact power values for these components are varying and depend on the specific architecture considered; however, only approximations are needed. Such an approximation has been proposed by Butts and Sohi [11], modeling the static power consumption due to leakage power as $P_{\text{stat}} = V \cdot N \cdot k_{\text{design}} \cdot I_{\text{leak}}$, where V is the supply voltage, N is the number of transistors, k_{design} is a design dependent parameter, and I_{leak} is a technology-dependent parameter.

For DVFS processors, we are interested in investigating the dependence of the power consumption on the operational frequency, which can be scaled within a predefined interval $[f_{\min}, f_{\max}]$. The scaling can be expressed by a dimensionless scaling factor $s \geq 1$ which describes a smaller frequency $\tilde{f} < f_{\max}$ as $\tilde{f} = f_{\max}/s$. The following functional dependencies have to be considered: the frequency f depends linearly on the supply voltage V, i.e., $V = \beta \cdot f$ with some appropriate constant β. Thus, the dependence of the dynamic power consumption on the frequency f can be expressed as $P_{\text{dyn}}(f) = \gamma \cdot f^3$ with $\gamma = \alpha \cdot C_{\text{L}} \cdot \beta^2$ or with the corresponding scaling factor s as $P_{\text{dyn}}(s) = s^{-3} \cdot P_{\text{dyn}}(1)$ where $P_{\text{dyn}}(1)$ is the dynamic power consumption in the un-scaled case. This means that the dynamic power increases cubically with the operational frequency, which can be used to study the change of the dynamic power consumption with respect to varying frequency values. Using $V = \beta \cdot f$ for the static power consumption P_{stat} leads to a linear dependence of the static power on f, i.e., $P_{\text{stat}}(f) = \delta \cdot f$ with $\delta = N \cdot k_{\text{design}} \cdot I_{\text{leak}} \cdot \beta$ or $P_{\text{stat}}(s) = s \cdot P_{\text{stat}}(1)$ where $P_{\text{stat}}(1)$ is the static power consumption in the un-scaled case.

Reducing the operational frequency of a processor by a scaling factor of s usually decreases the power consumption; however, it also increases the execution time $T_P(1)$ of a program P by the same factor compared to an un-scaled execution, i.e., $T_P(s) = s \cdot T_P(1)$. This has to be taken into consideration for the time integration to compute the energy consumption $E_P(s) = (P_{\text{dyn}}(s) + P_{\text{stat}}(s)) \cdot s \cdot T_P(1) = (s^{-3} \cdot P_{\text{dyn}}(1) + s \cdot P_{\text{stat}}(1)) \cdot s \cdot T_P(1)$. Runtime experiments on an Intel Sandy Bridge show that the linear dependence of the runtime on the frequency scaling factor can more or less be observed. However, for smaller frequencies the runtime increases more than linearly, see Figs. 8 and 9. This effect is even stronger for the Core i7 Haswell architecture, see Figs. 10 and 11, since this architecture supports smaller frequencies.

4.2 Heuristic energy model

The second energy model considered in this article is derived by an approximation approach based on the discrete energy data measured. This model considers the entire power consumption of the CPU processing one of the benchmarks and uses data fitting to derive a closed formula. Thus, the derivation of this energy model applies the opposite approach compared to the model in Sect. 4.1: it starts with the discrete data from the experimental setting and uses curve fitting methods with the purpose of getting a continuous approximation of the discrete data. The heuristic model described in the following is new and has not been used before.

Due to the power data that we have collected in our experimental setup, e.g. given in Figs. 10 and 11, we assume an almost linear dependence of the power from the frequency f with unknown parameters, i.e. we assume the power to be approximated by $P_{\text{total}}(f) = a + bf^{1+\epsilon}$ with parameters a and b to be determined by curve fitting. Although these parameters do not correspond to physical constants, but are the result of a mathematical calculation, the parameter a can be interpreted as a static part of the power consumption that does not change with the frequency, whereas b captures a dynamic part of the power consumption that increases with the operational frequency of the CPU. For the parameter ϵ, several fixed values can be used so that the parameters

a and b can be determined by applying the least squares method. The values of a and b reflect the characteristics of the specific CPU. However, different benchmarks may have different values for the parameters a and b due to their specific computational and memory access behavior, leading to a different usage of the processor resources and different power requirements. The results of the curve fitting are included in the next section reporting the comparison of the predicted and measured energy values.

5 Evaluation of the energy models

In this section, we investigate to which extent the energy models presented in the last section are able to capture the power and energy consumption of standard benchmark programs on recent processors. The emphasis lies on a modeling of the energy consumption as a function of the frequency in the frequency range provided by the processor. For those frequencies, we quantitatively and qualitatively compare the measured energy values as reported in Sect. 3 with energy values provided by the models from Sect. 4. Section 5.1 considers an application-specific energy modeling and Sect. 5.2 addresses an application-independent modeling and discusses the differences.

5.1 Application-specific energy modeling

The physical energy model based on frequency scaling from Sect. 4.1 contains the parameters γ in the equation for the dynamic power and δ in the equation for the static power consumption. The values for γ and δ have been determined by curve fitting with the least squares method using the measured power values for the different frequencies for each individual application of the SPEC CPU2006 benchmark suite. The resulting parameters γ and δ are application-specific and can be used for a modeling of the energy consumption of the specific application as described in Sect. 4.1.

For the heuristic model from Sect. 4.2, $\epsilon = 0.2$ has been used. This value has been determined by experiments which have shown that this value is well suited for all applications investigated. The model again contains parameters a and b, and as just described, we have applied curve fitting to determine these parameter values for the individual application of the SPEC CPU2006 benchmark suite. Based on these application-specific parameters, Fig. 12 compares the measured and predicted energy consumption for the different SPEC CPU2006 benchmarks for the Sandy Bridge architecture using selected frequencies. The smallest and the largest frequencies as well as a medium frequency have been selected for the diagrams. The entry *predicted 1* corresponds to the physical energy model from Sect. 4.1, and the entry *predicted 2* corresponds to the heuristic energy model from Sect. 4.2. Figure 13 shows the same comparison for the Haswell architecture and also includes a comparison with an application-independent modeling to be discussed in the next subsection.

From Figs. 12 and 13 it can be observed that for the application-specific modeling both energy models are well suited to describe the energy consumption of most benchmark programs and both architectures. The correspondence between modeled and measured energy consumption is especially good for medium and large frequencies, as shown for frequencies $f = 2.5$ GHz and $f = 3.4$ GHz. Similar results are

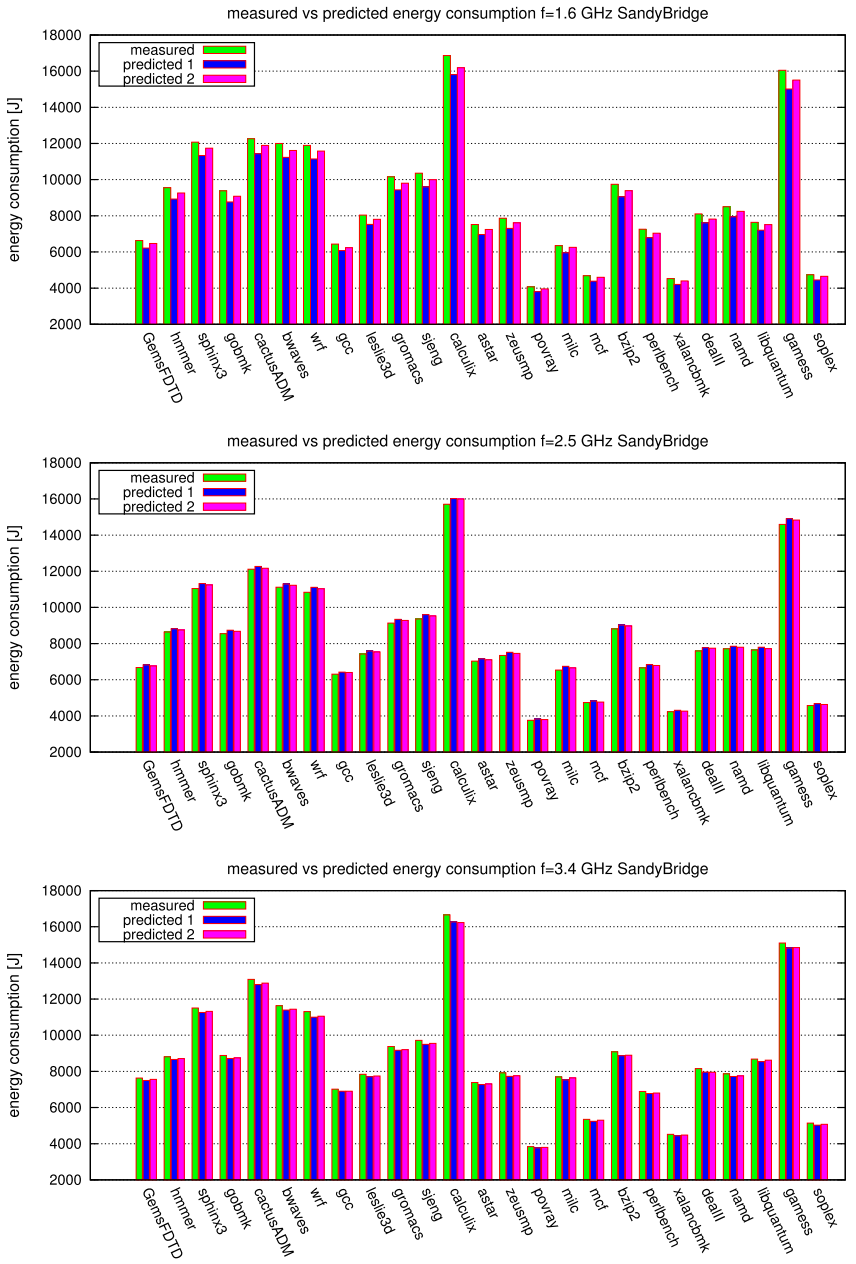

Fig. 12 Intel Core i7 Sandy Bridge processor: comparison of measured and predicted energy consumption of the SPEC CPU2006 integer and floating-point benchmarks for the frequencies $f = 1.6$ GHz (*top*), $f = 2.5$ GHz (*middle*), and $f = 3.4$ GHz (*bottom*)

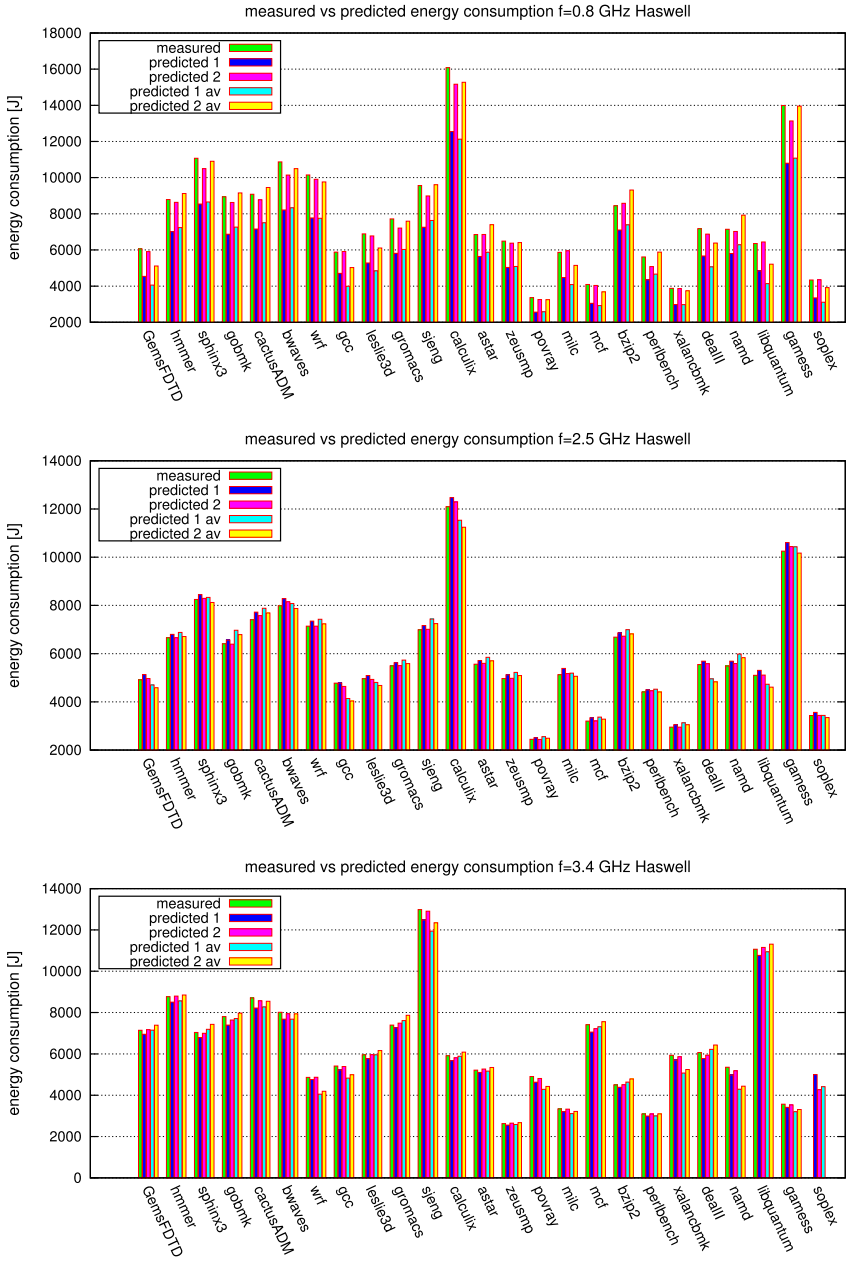

Fig. 13 Intel Core i7 Haswell processor: comparison of measured and predicted energy consumption of the SPEC CPU2006 integer and floating-point benchmarks for the frequencies $f = 0.8$ GHz (*top*), $f = 2.5$ GHz (*middle*), and $f = 3.4$ GHz (*bottom*)

obtained for other frequencies between these frequency values. The deviations usually lie below 10 %.

The comparison of the modeling quality using the two different models shows that for the Sandy Bridge processor, see Fig. 12, there is no significant difference between the energy values provided by the two models. On the Haswell architecture, larger deviations between the measured and predicted values can be observed for smaller frequencies, when the physical energy model from Sect. 4.1 is used, see the results for frequency $f = 0.8$ GHz in Fig. 13 (top) as example. These deviations are much smaller for the heuristic energy model. For small frequencies, the physical energy model systematically underestimates the energy consumption for both the Sandy Bridge and the Haswell processor. The reason for this underestimation is assumed to be an underestimation of the static power consumption when using $P_{\text{stat}}(f) = \delta \cdot f$. Other energy models [12,13] propose to use $P_{\text{stat}}(f) =$ constant for all frequencies, but this would lead to a significant overestimation of the total energy consumption for small frequencies (not shown in a figure), leading to even bigger deviations from the measured energy consumption. In contrast, the heuristic energy model captures the dynamic and the static energy consumption together as one physical entity in one formula, and this seems to be better suited for the whole range of frequencies. Thus, the heuristic model provides energy values that are better suited for a prediction than those provided by the physical energy model from Sect. 4.1.

Averaging over all frequencies and benchmarks, the average percentage difference between the measured energy values and the predicted values using the heuristic model with application-specific parameters is 1.8 % with a minimum difference of 0.9 % and a maximum difference of 6.0 %. Using the physical model, the average percentage difference is 5.6 % with a minimum difference of 4.4 % and a maximum difference of 10.7 %.

5.2 Application-independent energy modeling

One goal for modeling the energy consumption values is to provide a prediction model for the energy consumption of application programs. Such a model suitable as prediction model has to be easy to use and has to be application independent. Both the models introduced in Sect. 4 are represented by energy functions that are quite easy to evaluate. So both models meet the goal of simplicity. However, so far we have performed an application-specific modeling due to the parameters γ, δ, a, and b, respectively. An application-independent model requires parameters which are independent from a specific benchmark application but may depend on the processor architecture. In this subsection, we propose to use the average of the parameter values γ and δ or a and b over all benchmarks executed on the same processor.

For the different programs of the SPEC CPU2006 benchmark suite, the values of the parameters γ and δ resulting from the physical energy model are quite close to each other for most of the benchmarks on the same architecture; their difference is typically below 10 %. Thus, using the average values of the parameters also leads to a good correspondence between measured and modeled values. This represents an energy modeling that is independent from the specific application but is still architecture

dependent. For the different architectures, different values for the parameters γ and δ result.

The modeling with the average parameter values is additionally included in Fig. 13. The entry "predicted 1 av" corresponds to the modeling with the physical energy model from Sect. 4.1 using the average values for the parameters γ and δ taken over all SPEC CPU2006 applications on the Haswell processor. Similarly, the entry "predicted 2 av" corresponds to the modeling with the heuristic model using the average values for the parameters a and b. Using the average parameter values for the modeling leads to energy values that are slightly less close to the measured energy values than for the application-specific modeling, but the deviations are still acceptable. However, in some cases, especially for smaller frequencies, using the average parameter values for the modeling even leads to better predictions than the application-specific modeling. This is due to the strongly convex behavior of the measured energy values for frequencies between 0.8 and 1.6 GHz, see Figs. 10 and 11, which indicate a high overhead of the runtime even though the power function remains linear. The architectural effects are difficult to capture in one continuous functional pattern. However, the modeling with a piecewise continuous function would result in a too complex prediction model. For some benchmarks, the averaging of the parameters seems to level off extreme values. The heuristic energy model using the average parameter values for the modeling leads to slightly better predictions than the physical energy model with the average parameter values for the modeling.

From these observations it can be concluded that both energy models considered, although different, are able to capture the energy consumption and its dependence on the operational frequency with reasonable accuracy for most situations. Thus, both energy models can be used for an a priori prediction of the energy consumption as it is needed when solving problems such as optimization or scheduling. Compared to the physical model, the heuristic model leads to better predictions in most situations and should therefore be preferred. This is due to a better modeling of the leakage power, which may not be best represented by the linear term in the physical model. Other models proposed in the literature use constant, quadratic, or even exponential terms for the leakage power consumption. However, these models are usually not validate for a wide range of application programs.

6 Related work

Today, power-management mechanisms are integrated in computer systems of almost every size and class, from handheld devices to large servers [14]. Correspondingly, the analysis, measurement, or simulation of power and energy consumption is an active research area. An important feature is the DVFS technique, which trades off performance for power consumption by lowering the operating voltage and frequency if this is possible, see [4,13,15] for an overview. Frequency scaling due to the DVFS technique has been applied in practical programming as well as in theoretical investigations, optimizing or minimizing the energy consumption under certain constraints.

Approaches to determine the frequency scaling factor that minimizes the total CPU energy consumption by taking both the dynamic power and the leakage power into

consideration have been discussed in [13,16,17] for sequential programs. Voltage scaling has also be considered in [18].

Energy simulation models are often used in approaches to minimize the energy consumption by scheduling algorithms or other heuristics. Algorithmic research on speed scaling processors and related scheduling algorithms using the total energy consumption as objective function has been initiated by the article [19]. Many different scenarios and algorithms have been investigated in the literature, see [20–22] for an overview. For example, theoretical foundations of scheduling algorithms in a setting with dynamic speed scaling processors are investigated in [23], considering the scheduling of n jobs on m identical variable speed processors working in parallel, where each job is specified by a release date, a deadline, and a processing volume. Different scenarios concerning the job size, release dates and deadlines are considered and approximation algorithms for the resulting NP-hard scheduling problems are presented. In most of the articles in this research line, the emphasis lies on a theoretical investigation of the approximation algorithm derived and no simulations or measurements on real hardware systems are provided.

A comparison of energy measurements using PowerPack and RAPL (accessed via PAPI) for dense linear algebra algorithms is given in [1], showing that the RAPL measurements are a good alternative to physical power-meters. However, frequency scaling has not been taken into consideration in [1]. The power management architecture of the Sandy Bridge processor is described in [2] and includes a comparison of the actual measured power and the so-called architectural power-meter which predicts the active power consumption. The dependence of the power and the frequency is not directly investigated nor is the modeling with a theoretical power model included. An approach using RAPL for memory power estimation is discussed in [24]. The correspondence with simulation methods and prediction possibilities has also not been considered in the publication. A detailed comparison of power measurement techniques including RAPL has been given in [25]. However, the DVFS features with a variation of the frequency and the resulting impact on the power are not investigated. The performance API PAPI now also covers the report of energy and power values based on RAPL assuming that the code is previously instrumented, see [26]. The PAPI approach corresponds to the software approach accessing hardware counters in our work. [26] also lists state-of-the-art measurement techniques based on power-meters and hardware counters, but does not provide a comparison.

Performance prediction for DVFS processors is addressed in [27] with an emphasis on green supercomputing. The energy consumption of parallel algorithms for shared memory architectures based on the parallel external memory (PEM) model [28] has been discussed in [29]. [30] proposes a system-level iso-energy-efficiency model to analyze, evaluate and predict energy-performance of data-intensive parallel applications running on cluster systems. Based on measurements from smaller configurations, the power performance for larger systems with increasing number of nodes is predicted and analyzed. As example applications, the FT, EP, and CG benchmarks of the NAS Parallel Benchmarks are used. A scalability analysis shows a good correspondence of the predictions of the model. The interaction between the parallel execution time and the energy consumption is considered in [31] by partitioning a parallel algorithm into sequential and parallel regions and computing optimal frequencies for these regions.

Approaches for an energy complexity metric are discussed in [32]. The application of a physical energy model and its usage for the scheduling of sequential tasks in fork-join patterns with energy as objective function have been addressed in [33]; the energy-based scheduling of parallel tasks has been considered in [34].

In the domain of real-time scheduling, many techniques for utilizing available waiting times based on DVFS have been considered, see, e.g., [35–37]. These approaches are usually based on heuristics and are not based on an analytical model as presented in this work. The effects of dynamic concurrency throttling (DCT) and DVFS in the context of a hybrid MPI/OpenMP programming model are considered in [38]. In particular, frequency selection is formulated as a variant of the 0–1 knapsack problem and dynamic programming is used to compute an approximation.

7 Conclusions

In this article, we have investigated and evaluated measurement methods providing energy consumption values for recent DVFS processors. The article reports the experimental setting for the hardware as well as for the software measurement approach and presents the resulting measurement data. The CPUs investigated are the Intel Core i7 Sandy Bridge, Ivy Bridge and Haswell architectures. The application codes measured are the SPEC CPU2006 benchmarks. The analysis of the energy data gathered with the power-meter based and the RAPL-based energy measurements shows a good correspondence across all frequency values, CPUs and benchmarks, leading to the insight that both measurement methods are suitable and interchangeable. We consider this result to be especially valuable for research projects concerned with bigger applications as well as larger machines based on these recent CPUs. A second result of our investigations is that physical and heuristic energy models are well suited for an early prediction of the energy consumption of application programs. Thus, these energy models can be used as a basis for simulation approaches of the energy consumption of application programs.

Acknowledgments This work was supported by the federal Cluster of Excellence EXC 1075 “MERGE technologies for Functional Lightweight Structures” and the research grant RU591-10/2, both supported by the German Research Foundation (DFG).

References

1. Dongarra J, Ltaief H, Luszczek P, Weaver V (2012) Energy footprint of advanced dense numerical linear algebra using tile algorithms on multicore architectures. In: Proceedings of 2nd international conference on cloud and green computing (CGC). IEEE, pp 274–281
2. Rotem E, Naveh A, Ananthakrishnan A, Rajwan D, Weissmann E (2012) Power-management architecture of the intel microarchitecture code-named sandy bridge. IEEE Micro 32(2):20–27. doi:10.1109/MM.2012.12
3. Intel (2011) Intel 64 and IA-32 Architecture Software Developer’s Manual, System Programming Guide
4. Anshumali K, Chappell T, Gomes W, Miller J, Kurd N, Kumar R (2010) Circuit and process innovations to enable high-performance, and power and area efficiency on the Nehalem and Westmere family of Intel processors. Intel Technol J 14:104–127

5. Ge R, Feng X, Song S, Chang HC, Li D, Cameron K (2010) PowerPack: energy profiling and analysis of high-performance systems and applications. IEEE Trans Parallel Distrib Syst 21(5):658–671
6. LabVIEW. LabVIEW Measurement data format. http://www.ni.com/white-paper/4139/en
7. Treibig J, Hager G, Wellein G (2010) LIKWID: a lightweight performance-oriented tool suite for x86 multicore environments. In: 39th international conference on parallel processing workshops, ICPP '10. IEEE Computer Society, pp 207–216
8. Hennessy J, Patterson D (2012) Computer architecture—a quantitative approach, 5th edn. Morgan Kaufmann
9. Chen H, Shi W (2012) Power measurement and profiling. In: Ahmad I, Ranka S (eds) Handbook of energy-aware and green computing. CRC Press, pp 649–674
10. Kaxiras S, Martonosi M (2008) Computer architecture techniques for power-efficiency. Morgan & Claypool Publishers
11. Butts J, Sohi G (2000) A static power model for architects. In: Proceedings of the 33rd international symposium on microarchitecture (MICRO-33)
12. Lee Y, Zomaya A (2009) Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: CCGRID '09: Proceedings of the 2009 9th IEEE/ACM international symposium on cluster computing and the grid. IEEE Computer Society, pp 92–99
13. Zhuo J, Chakrabarti C (2008) Energy-efficient dynamic task scheduling algorithms for DVS systems. ACM Trans Embed Comput Syst 7(2):1–25
14. Saxe E (2010) Power-efficient software. Commun ACM 53(2):44–48. doi:10.1145/1646353.1646370
15. Usman S, Khan S, Khan S (2013) A comparative study of voltage/frequency scaling in NoC. In: 2013 IEEE international conference on electro/information technology (EIT), pp 1–5. doi:10.1109/EIT.2013.6632716
16. Irani S, Shukla S, Gupta R (2007) Algorithms for power savings. ACM Trans Algorithms 3(4):41 (2007). doi:10.1145/1290672.1290678
17. Jejurikar R, Pereira C, Gupta R (2004) Leakage aware dynamic voltage scaling for real-time embedded systems. In: DAC '04: proceedings of the 41st annual design automation conference. ACM, pp 275–280
18. Kim T (2012) Power saving by task-level dynamic voltage scaling: a theoretical aspect. In: Ahmad I, Ranka S (eds) Handbook of energy-aware and green computing. CRC Press, pp 361–383
19. Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. In: Proceedings of the 36th annual symposium on foundations of computer science, FOCS '95. IEEE Computer Society, Washington, DC, p 374. http://dl.acm.org/citation.cfm?id=795662.796264
20. Albers S (2010) Energy-efficient algorithms. Commun ACM 53(5):86–96. doi:10.1145/1735223.1735245
21. Bansal N, Kimbrel T, Pruhs K (2007) Speed scaling to manage energy and temperature. J ACM 54(1):3:1–3:39. doi:10.1145/1206035.1206038
22. Chrobak M (2012) Algorithmic aspects of energy-efficient computing. In: Ahmad I, Ranka S (eds) Handbook of energy-aware and green computing. CRC Press, pp 311–329
23. Albers S, Müller F, Schmelzer S (2007) Speed scaling on parallel processors. In: Proceedings of the 19th annual ACM symposium on parallel algorithms and architectures, SPAA '07. ACM, New York, pp 289–298. doi:10.1145/1248377.1248424
24. David H, Gorbatov E, Hanebutte U, Khanaa R, Le C (2010) RAPL: memory power estimation and capping. In: Proceedings of international symposium on low power electronics and design (ISLPED). ACM, pp 189–194
25. Hackenberg D, Ilsche T, Schöne R, Molka D, Schmidt M, Nagel W (2013) Power measurement techniques on standard compute nodes: a quantitative comparison. In: 2013 IEEE international symposium on performance analysis of systems and software, pp 194–204
26. Weaver V, Johnson M, Kasichayanula K, Ralph J, Luszczek P, Terpstra D, Moore S (2012) Measuring energy and power with PAPI. In: Proceedings of the ICPP workshop on power profiling and evaluation, workshop proc. of ICPP 2012. IEEE Computer Society, pp 262–268
27. Rountree B, Lowenthal DK, Schulz M, de Supinski BR (2011) Practical performance prediction under dynamic voltage frequency scaling. In: Proceedings of the 2011 international green computing conference and workshops, IGCC '11. IEEE Computer Society, Washington, DC, pp 1–8. doi:10.1109/IGCC.2011.6008553
28. Arge L, Goodrich M, Nelson M, Sitchinava N (2008) Fundamental parallel algorithms for private-cache chip multiprocessors. In: SPAA '08: proceedings of the 20th annual symposium on parallelism in algorithms and architectures. ACM, pp 197–206. doi:10.1145/1378533.1378573

29. Korthikanti V, Agha G (2010) Towards optimizing energy costs of algorithms for shared memory architectures. In: SPAA '10: proceedings of the 22nd ACM symposium on parallelism in algorithms and architectures. ACM, New York, pp 157–165. doi:10.1145/1810479.1810510
30. Song S, Su CY, Ge R, Vishnu A, Cameron K (2011) Iso-energy-efficiency: An approach to power-constrained parallel computation. In: Proceedings of the 25th IEEE international parallel and distributed processing symposium (IPDPS 11). IEEE
31. Cho S, Melhem R (2008) Corollaries to Amdahl's law for energy. IEEE Comput Archit Lett 7(1):25–28 (2008). doi:10.1109/L-CA.2007.18
32. Bingham B, Greenstreet M (2008) Computation with energy-time trade-offs: models, algorithms and lower-bounds. In: ISPA '08: proceedings of the 2008 IEEE international symposium on parallel and distributed processing with applications. IEEE Computer Society, pp 143–152. doi:10.1109/ISPA.2008.127
33. Rauber T, Rünger G (2012) Energy-aware execution of fork-join-based task parallelism. In: Proceedings of the 20th international symposium on modeling, analysis and simulation of computer and telecommunication systems (MASCOTS'12). IEEE
34. Rauber T, Rünger G (2012) Towards an energy model for modular parallel scientific applications. In: IEEE international conference on green computing and communications (GreenCom 2012). IEEE, pp 523–532. doi:10.1109/GreenCom.2012.79
35. Horvath T, Abdelzaher T, Skadron K, Liu X (2007) Dynamic voltage scaling in multitier web servers with end-to-end delay control. IEEE Trans Comput 56(4):444–458. doi:10.1109/TC.2007.1003
36. Mishra R, Rastogi N, Zhu D, Mossé D, Melhem R (2003) Energy aware scheduling for distributed real-time systems. In: IPDPS '03: proceedings of the 17th international symposium on parallel and distributed processing. IEEE Computer Society, p 21.2
37. Zhu D, Melhem R, Mossé D (2009) Energy efficient redundant configurations for real-time parallel reliable servers. Real Time Syst 41(3):195–221. doi:10.1007/s11241-009-9067-8
38. Li D, de Supinski B, Schulz M, Nikolopoulos D, Cameron K (2012) Strategies for energy efficient resource management of hybrid programming models. IEEE transaction on parallel and distributed systems