An Emulation-Based Real-Time Power Profiling Unit for Embedded Software

Andreas Genser\textsuperscript{1}, Christian Bachmann\textsuperscript{1}, Josef Haid\textsuperscript{2}, Christian Steger\textsuperscript{1} and Reinhold Weiss\textsuperscript{1}

\textsuperscript{1}Institute for Technical Informatics, Graz University of Technology, Austria
\textsuperscript{2}Infineon Technologies Austria AG, Design Center Graz, Austria
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
josef.haid@infineon.com

Abstract—The power consumption of battery-powered and energy-scavenging devices has become a major design metric for embedded systems. Increasingly complex software applications as well as rising demands in operating times while having restricted power budgets make power-aware system design indispensable. In this paper we present an emulation-based power profiling approach allowing for real-time power analysis of embedded systems. Power saving potential as well as power-critical events can be identified in much less time compared to power simulations. Hence, the designer can take countermeasures already in early design stages, which enhances development efficiency and decreases time-to-market. Accuracies achieved for a deep sub-micron smart-card controller are greater than 90\% compared to gate-level simulations.

I. INTRODUCTION

Rising complexity of embedded software applications and the advance in processing power available in embedded systems require power analysis techniques to identify power saving potential. Furthermore, the detection of power-critical events, such as power peaks, which can affect system stability of energy-scavenging devices (e.g. contact-less smart-cards) is of great importance.

Among all abstraction layers the greatest power reduction potential can be identified on the application layer [1]. To enable the design of power-efficient software applications, power consumption feedback to the software designer should be available already at early design stages. However, commercially available power estimation and analysis tools are often operating on low abstraction layers, which are usually not available to the software designer. Moreover, low-level power simulations lead to extensive run-times. This makes power simulations for complex designs unfeasible.

Functional hardware emulation by means of prototyping platforms, such as FPGA-boards, has become a widespread technique for functional verification. Power information, however, is still in many cases gathered by power simulators. In this work, which is part of the PowerHouse\textsuperscript{3} project, we propose an emulation-based real-time power profiling approach to circumvent this limitation. A given design augmented with power estimation hardware allows for obtaining power alongside functional characteristics in real-time. Power saving potential or power peaks can hence be detected earlier in the design cycle, which normally is not feasible before the design is available in silicon and actual physical measurements can be carried out.

By coupling this approach with a software development environment, valuable power information can be transferred to the software designer. This concept is depicted in Fig. 1. The FPGA-platform collects functional verification and power characteristics information, which is transmitted to a host computer. These information can be evaluated and visualized in a software development environment.

This paper is structured as follows. Section II provides information on previous work on power profiling. Section III briefly shows our research contributions. In Section IV the design of the real-time power profiling unit is discussed. Section V outlines a case-study applying the concepts developed in this work to a contact-less deep sub-micron smart-card controller and finally, conclusions drawn from the current work are summarized in Section VI.

II. RELATED WORK ON POWER PROFILING

Embedded software application power profiling can be categorized in (i) measurement-based and (ii) estimation-based methods.

Measurement-based methods are performed by taking actual physical measurements. This yields high accuracy compared to other approaches but requires additional measurement-equipment.

In contrast, power profiling by means of estimation methods is often based on power modelling. These techniques are usually less accurate but provide greater flexibility, since

\begin{figure}[h]
  \centering
  \includegraphics[width=0.8\textwidth]{figure1.png}
  \caption{Overview of an emulation-based power profiling approach for embedded systems comprising host computer interaction and visualization.}
\end{figure}

\textsuperscript{3}Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.
also power consumption for sub-modules of the system can be derived. In the following we compare ongoing research activities in the field of power profiling.

A. Measurement-Based Methods

In [2], PowerScope an energy profiling tool for mobile applications is introduced. The system’s current consumption is automatically measured during run-time by a digital multimeter. Measurement data are collected for later analysis on a host computer.

An oscilloscope measurement-based profiling technique is proposed by Texas Instruments in [3]. The current drawn by a DSP system is profiled and results are visualized on a host computer in TI’s software development environment.

B. Estimation-Based Methods

Power profiling by means of estimation techniques can be subdivided into (i) simulation-based and (ii) hardware-accelerated approaches.

Simulation-based power estimation executes programs on simulators to obtain circuit activity information. Power values are acquired using these information. In hardware-accelerated power estimation approaches, power information is derived from power models, which are implemented in hardware.

Estimation techniques can be employed on various levels of abstraction resulting in different estimation accuracies. Moreover, the degree of abstraction influences simulation times for simulation-based approaches and hardware-effort for hardware-accelerated methods. Real-time power estimation, however, is limited to hardware-accelerated estimation techniques.

Commercially available power estimation tools (e.g. [4]) operate on low abstraction levels, such as gate- or register-transfer level (RTL). Achievable estimation accuracies are high, while extensive simulation times render power estimation of elaborate applications unfeasible. On top of this, low-level simulators are often not available to software designers. Therefore, attempts to estimate the system’s power consumption on a higher level of abstraction are carried out.

A simulation-based approach employing power models for instruction-level power estimations is proposed by Tiwari et al. in [5]. It allows for power and energy consumption estimation for given applications. The underlying power model considers the power consumption during instruction execution (i.e. base costs) and power consumption during the transition between instructions (i.e. circuit state overhead costs). In [6], Sami et al. consider additional microarchitectural effects to enhance the accuracy of instruction-level power estimation based on a pipeline-aware power model for Very-Long-Instruction-Word (VLIW) architectures. A co-simulation based power estimation technique is introduced by Lajolo et al. in [7]. This approach for System-On-Chips (SoCs) works on multiple abstraction levels. In principle, power estimation is performed on system level, while for refinement purposes and accuracy enhancements various components are co-simulated on lower levels of abstraction. Countermeasures against high simulation times are caching, statistical sampling and macro-modelling. A simulation framework for system-level SoC power estimation is introduced by Lee in [8]. This approach is based on power models developed for the processor, memories and custom IP blocks. Power values derived are provided cycle-accurately to the designer in a dedicated profile-viewer.

Hardware-accelerated power estimation techniques are performed by augmenting the given system with dedicated hardware blocks. A power characterization process performed beforehand determines power values, which are mapped towards corresponding power states. For example, hardware events (e.g. CPU idle/run states, memory read/write states, etc.) are representatives of such power states. For energy accounting, existing power estimation hardware can be extended to power state counters. In [9], Bellosa gathers information by means of hardware event counters to derive thread-specific energy information for operating systems. Joseph et al. obtain the power consumption of a system by exploiting existing hardware-performance counters of a microcontroller [10]. A power macro-model based coprocessor approach for energy accounting is proposed in [11]. Energy events identified by energy sensors are tracked by a central controller. The additional power estimation hardware requires extra chip area but yields also a speed-up compared to simulation-based approaches.

Power emulation represents a special case of hardware-accelerated power estimation. FPGA-boards can be used as a typical prototyping platform to emulate not only the functional system behavior but also its power consumption. A given system comprising power estimation hardware is mapped onto an FPGA-platform. Functional verification and power estimation can be performed in real-time even before the silicon implementation of the system is available.

An overview of the power emulation principle is presented in [12]. Run-time improvements by power estimation hardware-acceleration of about 10x to 500x compared to commercial power estimation tools are achieved. Strategies to minimize the hardware overhead introduced by power estimation are proposed. In [13], this approach is extended to a hybrid power estimation methodology for complex SoCs. This framework combines simulation and emulation techniques, which significantly reduce power analysis times. In [14], the power consumption of processor cores is estimated employing power emulation to guide process migration between cores.

III. CONTRIBUTIONS

Power profiling by means of physical measurements is typically very coarse-grained and limited to the entire chip due to chip integration and packaging. Moreover, the final chip is not available at early design stages.

Simulation-based power profiling techniques can be employed at the beginning of the design cycle. However, they are rendered unfeasible for complex applications due to extensive simulation times. To encourage the software designer to consider power aspects at early design stages, we provide a real-time emulation-based power profiling approach. Power
information is delivered to the software designer before silicon is available by utilizing an FPGA prototyping platform comprising power estimation hardware. Expensive redesigns caused by ‘power bugs’ can be avoided, which helps to decrease time-to-market (see Fig. 2).

The main goals of this work can be defined as follows:

- Power-efficient software application design
- Power-critical event detection
- Reduce development times

IV. DESIGN OF A REAL-TIME POWER PROFILING UNIT

Estimation-based power profiling methods derive power information by exploiting power models. The abstraction layer on which these models are set up determines complexity and accuracy. Low-level models established on transistor- or gate-level are complex and therefore not suitable for emulation-based power profiling. In contrast, on a higher abstraction layer only main system components (e.g., CPU, memory, coprocessor, etc.) are taken into account. This leads to more compact models, hence real-time power profiling with moderate area increases is only feasible following this approach.

A. Power Model

Power models on a high level of abstraction are often based on linear regression methods. Details can be obtained in [15] and implementations are discussed in [5], [16]. A linear regression model can be given as

\[ y = \sum_{i=0}^{n-1} c_i x_i + \epsilon. \]  

(1)

\[ x = [x_1, x_2, ... x_{n-1}] \]

vectors give the vector of model parameters. \( x_i \) represent system states, such as CPU modes (e.g. idle, run) or memory accesses (e.g. read, write) etc. The vector of model coefficients is given as \( c = [c_1, c_2, ... c_{n-1}]^T \). Each model coefficient \( c_i \) contains power information and has to be determined during a preliminary power model characterization process. The linear combination of model parameters \( x \) and model coefficients \( c \) form the power estimate \( y \). The deviation between the real power value and its estimate \( y \) is stated by \( \epsilon \) (i.e. the estimation error).

B. Power Characterization Process

Typically a linear regression model can be designed in three major steps.

(i) Selection of model parameters. The choice of model parameters directly influences the model’s accuracy and is therefore of great importance. In addition, the cross-correlation between model parameters reflects the amount of redundancy in the model. This metric helps to keep a model as small as possible and thus efficient.

(ii) Selection of the training-set. The training-set is based on \( m \) power measurements for a number of \( m \) vectors \( x^i \), each of which containing \( n \) model parameters

\[ x^i = [x_{i0}, x_{i1}, ... x_{i(n-1)}] \text{ for } 0 \leq i \leq m - 1. \]  

(2)

Vectors \( x^i \) can be combined to the matrix

\[ X = [x^0, x^1, ... x^{m-1}]^T. \]  

(3)

Power values \( y \) acquired for corresponding vectors \( x^i \) can be expressed as

\[ y = [y_0, y_1, ... y_{m-1}]^T. \]  

(4)

Finally, \( X \) and \( y \) define the training-set and can be given as the tuple \( T \) in (5).

\[ T = (y, X) \]  

(5)

The linear regression model given in (1) can also be written in matrix-form. This is depicted in (6).

\[ y = Xc \]  

(6)

Vectors \( x^i \) in \( X \) are derived from test applications (benchmarks) on a given embedded system and corresponding power values \( y \) are determined by physical power measurements or gate-level simulations.

(iii) Least squares fit method. The number of elements in the training-set \( T \) is usually much higher than the number of model parameters \( c \). This implies that the number of rows in \( y \) and the number of columns in \( X \) are higher than actually required to solve the linear system of equations in (6). Hence, the system is overdetermined and no exact solution exists. To overcome this issue model parameters are determined while minimizing the square error by using the least squares fit method.

The above steps can be carried out iteratively. If the model’s accuracy does not meet the requirements, a model refinement by accounting more low-level information can be applied.

C. Power Emulation Architecture

The power emulation (PE) architecture that integrates the power model in hardware is illustrated in Fig. 3.

Power sensors are employed to track state information of system modules. For accuracy purposes also lower abstraction layer information can be considered (e.g. state information of functional units of the CPU). State vector and power...
information are stored in a software-configurable table. These state information is mapped towards power values using a table-lookup approach. Fig. 4 depicts the principle structure of a power sensor module.

Each of a number of $k$ power sensors covers $l$ system states and contributes to the entire power model as expressed in (7). The PE-architecture delivers power information each cycle, hence time-dependency $t$ is introduced in the following equations to account for power values estimated at different points in time.

$$y_{j,i}(t) = c_i \cdot x_i(t) \text{ for } 0 \leq i \leq l - 1 \land 0 \leq j \leq k - 1$$

(7)

16-bit registers are provided to configure the power sensors with the power coefficients information obtained from $c$. It is worth noting that this power table can also be reconfigured during program run-time. This enables the masking of system modules, allowing the tracking of the power consumption of single sub-modules.

The power estimation unit accumulates 16-bit power sensor outputs according to (8). This constitutes an instantaneous, cycle-accurate up to 32-bit wide power estimate $y(t)$ for the overall system. The entirety of power sensors comprising the power estimation unit represent the power model established in hardware (see equality in (8)).

$$y(t) = \sum_{j=0}^{k-1} \sum_{i=0}^{l-1} y_{j,i}(t) = \sum_{i=0}^{n-1} c_i \cdot x_i(t)$$

(8)

Further post-processing is applied by the averaging module, which allows for smoothing and de-noising of a sequence of power values. This is enabled by a configurable moving average filter as shown in (9). Filtering properties can be changed by adjusting $N$.

$$y_{avg} = \frac{1}{N} \sum_{j=0}^{N-1} y(t - j)$$

(9)

The debug-trace generator unit captures power information of the power estimation unit and the averaging module. A debug-trace message is composed out of these data and is delivered to the host computer for evaluation and further processing.

D. Design Flow

Fig. 5 outlines the design flow of the emulation-based real-time power profiling approach. A synthesizable RTL-model of the target system is provided to perform the characterization process. After synthesis gate-level simulations based on benchmarking applications are performed and activity information as well as power profiles are acquired from value change dump (VCD) files. These information is fed to a power modelling process, deriving power model coefficients.

The target system RTL-models and the PE-architecture are merged to allow the generation of a single netlist. After downloading the netlist onto the FPGA-platform, power model coefficients determined beforehand are used to configure the power sensors for tailoring the PE-architecture to the given target system. Now applications of interest can be executed and real-time power profiles can be obtained.

E. System Set-Up

Power model coefficients obtained during the characterization process deliver configuration data for the power sensors. Listing 1 illustrates how to configure power sensors to tailor them to the power consumption of system modules. 16-bit registers are provided for this purpose.

```
// start of program

// configuration of power sensor1
PWRSEN0_STATE0 = 0x005A; //CPU run mode
PWRSEN0_STATE1 = 0x0011; //CPU halt mode
PWRSEN0_STATE2 = 0x0013; //CPU sleep mode

// configuration of power sensor2
PWRSEN1_STATE0 = 0x0013; //memory read
```
PWRSEN1_STATE1 = 0x001A; // memory write

...activate_power_emulation();
start_main_program();

Listing 1. Power emulation set-up, power sensor configuration

A variable number of system states for a variable number of system modules can be configured with power coefficients. Finally, power profiling is activated before normal application execution starts. The run-time overhead introduced due to power sensor configuration is negligible compared to the number of cycles executed by a typical application. A debug-trace containing power information is automatically generated and transferred to the host computer for power information evaluation and profile visualization without the interaction of the software designer.

If power analysis of sub-modules of the system is desired, the power sensor attached to the module of interest is configured as usual. The configuration of the remaining modules is skipped, so they do not contribute to the overall power consumption.

V. CASE STUDY: PROFILING OF POWER-CRITICAL SMART-CARD APPLICATIONS

Smart-card applications have been penetrating manifold market segments in the last years. Access control, electronic passport or payment are only a few out of many existing applications.

Smart-cards in general can be categorized in (i) contact-based and (ii) contact-less derivatives. Contact-based smart-cards are powered if inserted into a reader device, while contact-less systems consume power via an RF-field generated by the reader. Therefore, contact-less devices are subjected to stringent power limitations.

Fig. 6 illustrates the coupling of the reader device with the contact-less smart-card by a magnetic field $H$. A certain amount of power is transferred from the reader device to the smart-card at a time. The available power is limited, hence exceeding a maximum power limit due to power-peaks affects system stability and can cause malfunctions. This case-study demonstrates the capability of our emulation-based power profiling approach to support the software designer early in the development process to avoid such worst-case scenarios.

A. Smart-Card Architecture Overview

Fig. 7 depicts a typical contact-less smart-card system. It is based on a 16-bit pipelined cache architecture comprising volatile and non-volatile memories. A symmetric coprocessor (SCP) is included for Advanced Encryption Standard (AES) and Data Encryption Standard (DES) algorithm acceleration. Moreover peripherals, such as a UART for communication purposes, timers or a random number generator (RNG) are provided. System modules are powered by an externally generated RF-field. Energy is collected by an antenna system and power supply conditioning and stabilization by means of an analog front-end are carried out.

System modules that are major contributors to the overall power consumption are identified during a power characterization process. Moreover, available operating modes of each system module influencing the amount of power consumed are considered. A number of benchmarking applications to test many of these operating modes were applied. Based on the result of the characterization process, power model coefficients were obtained to configure the power sensors as shown in Listing 1. Table I summarizes system modules and corresponding operating modes relevant for the power model.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Mode(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>run, halt, sleep</td>
</tr>
<tr>
<td>Cache</td>
<td>read, write (hit, miss)</td>
</tr>
<tr>
<td>Memories</td>
<td>read, write</td>
</tr>
<tr>
<td>UART</td>
<td>read, write</td>
</tr>
<tr>
<td>Peripherals</td>
<td>on, off</td>
</tr>
<tr>
<td>SCP</td>
<td>Encryption: AES128/192/256, (Single-, Double-, Triple-) DES</td>
</tr>
<tr>
<td>SCP</td>
<td>Decryption: AES128/192/256, (Single-, Double-, Triple-) DES</td>
</tr>
</tbody>
</table>

B. Payment Application Profile Analysis

A typical application for smart-cards incorporating a symmetric cryptographic coprocessor is payment. Payment applications usually contain authentication procedures requiring...
cryptographic operations. Fig. 8 illustrates a typical power profile of a future payment application obtained with the emulation-based power profiling approach. The first power peak marks the power consumption of an AES computation, while the remaining slightly smaller and shorter power peaks result from DES computations.

We assume that the maximum available power provided by the RF-field is 0.9 as shown in Fig. 8 (Note that power values are normalized). Hence, the payment authentication process would fail due to power peaks caused by the SCP.

If the source of these power peaks is not obvious to the designer already at this step, the power profile can be decomposed into sub-modules by reconfiguring power sensors. Fig. 9 depicts power profile results for the CPU, memories, SCP and UART decomposed for sub-module power profile analysis. As a reference also the cumulated power profile is shown.

The major contributor to the power consumption in this example is the CPU with more than 60%. Memories (including cache) and the SCP account for the remaining power consumption. The UART is inactive in this application and therefore consumes no power. It can clearly be seen that the SCP’s power consumption causes the overall power profile to exceed the absolute maximum power of 0.9.

Various countermeasures could be taken to circumvent this issue. The AES algorithm could be implemented in software to avoid using the SCP. Another alternative is reducing the system clock frequency. Both solutions reduce the payment application’s speed, but ensure reliable operation. Fig. 10 shows the power profile when scaling down the system frequency from 33 MHz to 28 MHz. Power peaks are below 0.9, hence system operation is stable.

C. Accuracy of Emulation-based Power Profiling

Fig. 11 shows power profiles of the payment application. A comparison between the emulation-based power profiling result and gate-level power simulation profiles obtained with Magma Blastfusion 5.2.2 [17] are given. The relative error on average and the variance are 8.4% and 9.4%, respectively.

Accuracy considerations for other executed applications are summarized in Table II. The relative power error on average and the corresponding variance are given. Moreover, relative energy error values are depicted. For all tested applications relative power errors are less than 10% on average. Energy accounting reaches accuracies of greater than 93%.
D. Performance Evaluation

One of the major advantages of the emulation-based profiling approach is the capability to acquire power profiles in real-time. Power estimation takes no longer than application execution on the target-system. Hence, the power profile is available immediately after execution. Table III shows execution times of power profile simulations on a gate-level basis using Magma Blastfusion compared to the emulation-based approach. Extensively high simulation times are depicted compared to emulation run-times of a few hundreds of microseconds.

As illustrated in Table III, power emulation of the payment application takes 338 microseconds, whereas the power simulation is about 98 hours. Additional hardware introduced for power emulation contributes only to 1.5% to the overall system area.

It is obvious that the power simulation of complex applications is rendered unfeasible due to far too extensive simulation times. Therefore, power saving potential or power peaks leading to system failure as discussed in this section cannot be detected in an early design stage. By means of emulation-based power profiling, power estimates are available immediately and already when working with FPGA prototyping platforms. Countermeasures to circumvent power peaks causing system failures can be taken before the device is available on silicon and physical measurements are performed.

VI. CONCLUSION

Extensive run-times of power simulators render power analysis of increasingly complex embedded software applications unfeasible. The power profiling approach proposed in this work, delivers power information to the software designer in real-time. Moreover, by employing FPGA prototyping platforms, these power information is available already at early design stages. Emulation-based power profiling has proven to be an effective option to estimate a system’s power consumption, delivering power information with accuracies of 90% on average. This paves the way for power-efficient embedded software design and the capability to cope with power critical events more efficiently.