Nunez-Yanez, J. L., & Lore, G. (2013). Enabling accurate modeling of power and energy consumption in an ARM-based System-on-Chip. Microprocessors and Microsystems, 37(3), 319-332. 10.1016/j.micpro.2012.12.004

Enabling accurate modeling of power and energy consumption in an ARM-based System-on-Chip.

Jose Nunez-Yanez University of Bristol, Geza Lore ARM Ltd

Abstract: Motivated by the importance of energy consumption in mobile electronics this work describes a methodology developed at ARM for power modeling and energy estimation in complex System-on-Chips (SoCs). The approach is based on developing statistical power models for the system components using regression analysis and extends previous work that has mainly focused on microprocessor cores. The power models are derived from post-layout power-estimation data, after exploring the high-level activity space of each component. The models are then used to conduct an energy analysis based on realistic use cases including web browser benchmarks and multimedia algorithms running on a dual-core processor under Linux. The obtained results show the effects of different hardware configurations on power and energy for a given application and that system level energy consumption analysis can help the design team to make informed architectural trade-offs during the design process.
Keywords: energy modeling, power estimation, system-level modeling.

1. Introduction

Energy efficiency is one of the primary design constraints for mobile devices that need to operate autonomously for as long as possible [1]. The current rate of battery life improvement of around 5 % per year [2] means that the limited energy budget could delay the introduction of the future chips needed to support workloads whose complexity increases by one order of magnitude every five years [3]. Additionally, minimizing power consumption reduces the amount of heat dissipated requiring lower cost packaging, cooling solutions and increasing device reliability. Available studies show that a 10 degree increase in working temperature causes a 100% increase in failure rate [4].

The first action required to achieve the objective of minimizing energy or power consumption is to understand how the available energy budget is being used in the system [5]. This means that not only the main components such as the processor cores should be considered, but the whole system should be analyzed using real workloads as inputs to account for the dependency of power on the dynamic behaviour of applications. These analysis results should be available before the device has been fabricated and should be accurate enough to guide the

design team in the process of making architectural decisions leading to a solution superior in power and energy terms.

Fig 1 depicts the typical main components that can be found in a modern mobile phone. This paper focuses on the interaction of the application processor and the memory subsystem. These parts account for roughly 30% to 50% of the total device power budget in compute intensive applications such as media playing [5]. The power consumption of the LCD/backlight combination and radios are the two other main component drains in the system but they are considered to be beyond the scope of this work. With these constrains in mind the present work contributions are as follows:

1. The introduction of a power modeling methodology based on implementation data and regression analysis at the system level. The methodology enables the creation of accurate models that can then be stimulated from only sparse trace information collected during long application runs.

2. The analysis of power and energy consumption trends in a state-of-the-art multiprocessor architecture with realistic benchmarks including internet browsing and video coding running on a Linux operating system.

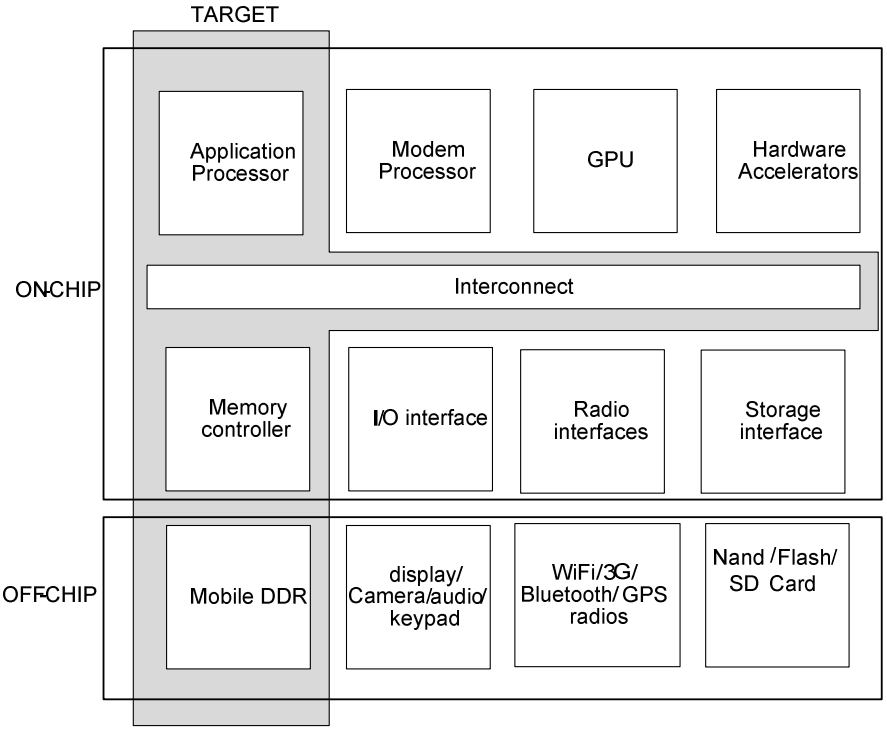

Fig. 1. High-level view of a mobile phone architecture

The rest of the paper is organized as follows. Section 2 positions this work in relation to existing work in the field of power modeling. Section 3 presents the power modeling

methodology in detail. Section 4 presents the target system investigated in this work centered around a dual-core processor. Section 5 summarizes the power models developed for the different components. Section 6 showcases the application of the methodology to the estimations of power and energy consumption for a set of benchmarks and real applications. Finally section 7 concludes this paper indicating its capabilities, limitations and stating future work.

2. Related work

This section makes a classification between research that considers power modeling at individual component level and system level which is the approach followed in this work.

2.1 Power modeling at the component level

This section reviews power modeling that considers independent components like the microprocessor core.

2.1.1 Microprocessor core

The component that has received most attention is naturally the CPU [6]. The standard approach consists of using existing cycle-level architectural simulators such as SimpleScalar [7] extended with tightly coupled power estimation capabilities. The power models added to the simulators are based on either analytical or empirical techniques [8]. Analytical techniques are useful for regular structures such as RAM-based structures (cache, register files, buffers etc). These analytical models suffer from inaccuracies since they cannot properly capture node capacitance that depends heavily on design layout. They are complemented with empirical models based on power analysis of structures expected to be reused, with data extracted from recently designed processors. This methodology is used in power modeling research tools such as Wattch[9], SimplePower[10], and PowerTimer [11].
Wattch [9] is a power-performance simulator widely used within the academic community. The base performance simulator is SimpleScalar and can be used to investigate the effects of cache organization, pipelining, multi-instruction issue, etc on power. The energy models used in Wattch are based on scaled power numbers obtained from published values or analytical equations for the regular structures. SimplePower [10] tries to improve accuracy by capturing power variations due to switching-activity in the processor logic blocks. Detailed simulation-based circuit energy is obtained for possible cycle-to-cycle transitions of the input

pins of the different subunits and the values are stored in look-up tables. A similar approach is followed in PowerTimer [11] that includes a hierarchical suite of energy functions that are refined as the design and simulation model evolves. This allows a progressive improvement of the estimation accuracy when the circuit data becomes available.

These solutions share the common characteristic that they are focused on the processor and that the methodology is based on obtaining power data for each of the individual microarchitectural components present in the microprocessor. The power data is stored in look up tables in which a different entry exists for each possible input transition. These tables can grow very large and Wattch uses a simple fixed-activity model. Wattch only tracks the number of accesses to a specific component and utilizes an average capacity value to estimate the power consumed. The number of functional units considered in the processor can become very large as shown in [2] that models the power for a single 8-bit carry-select adder decomposed into its individual constituent gates. This approach is expected to be accurate for individual components but scaling it to full processors is complicated, slows down the simulation speed considerably due to the number of components that is necessary to trace and has limited accuracy because, for example, it neglects the layout of these units in relation with the other units and this can introduce errors. The advantage of this fine level decomposition is that it allows the study of the power effects of replacing subunits, for example, using a different addition technique. In contrast, our processor model is simpler although it is tuned to the Cortex-A9 processor [12] considered in this work and a different model should be developed if a different processor will be used. In essence our approach focuses on system architectural changes rather than microarchitectural changes in the processor.

Another challenge also present in the above described techniques is that the low level switching activity needed to address the look up tables will not be generally available in simulators such as SimpleScalar that make a number of simplifications to make the simulation numerically efficient [6]. The speed efficiency of simulators such as SimpleScalar is obtained by abstracting away the microarchitectural details but these details are required to obtain an accurate power figure. The solution is to add back some of this complexity but this slows down the simulation significantly [6]. Overall the discrepancies among the power data obtained by these simulators can be considerable not only affecting the absolute power values but, more importantly, the power trends which could make the designer make the wrong decisions as established in [6].

More recently the work done in [13] targets the IBM POWER architecture using linear regression, focusing on how to select the activity measures suggesting a combination of experience and a mathematical systematic approach. The resulting power model for the POWER family of processors uses a total of 36 activity measures and is considerably more complex than the proposed power model for the cores in the current work. In our case the selection of activity measures is restricted to measures that are easily accessible in the analyzed IP and that are abstract enough so that they can be collected from alternative simulation models as well.

The application of linear regression to power modeling without explicit workload activity as done in this paper is investigated in [14]. PowerTimer is used with an out-of-order superscalar processor simulator [15] to obtain power estimates based on transistor-level power analysis and resource utilization statistics. The approach consists of monitoring the system utilization varying a set of parameters that include the L2 cache size, main memory latency, number of physical registers, while running the SPEC CPU2000 [16] benchmarks. This data is then used to build the power models using regression techniques. The resulting power models reduce the amount of simulation needed in the sense that they can predict power based on a number of selected parameters such as number of functional units, number of reservation stages, pipeline depth, number of physical registers, etc. Nevertheless, since there is strong dependence between power and workload a number of predictors are added to the power models to reflect processor activity such as L1 cache misses, branch misspredictions, etc requiring some simulation data.

All the methods discussed so far require low-level knowledge of the processor under investigation comprising circuit-level, gate-level or register transfer descriptions. Other work logs the number of instructions executed without considering microarchitectural effects. Techniques based on instruction-level power analysis [17] leave out important details such as pipeline stalls, cache misses, circuit state that affect switching activity etc. For example a modern out-of-order superscalar processor such as the one considered in this work expends a significant amount of energy extracting instruction level parallelism, register renaming, etc compared with the activation of functional units corresponding to the executed instructions. Alternatively, functional unit power analysis based on analytical expressions has also been developed that require no detailed knowledge of the processor circuitry [18] and is expected to increase accuracy compared with instruction-level. The basic idea is the distinction of functional blocks like the processing unit, instruction management unit, internal memory etc. Power is measured in the fabricated chip and arithmetic functions are developed

for each block to determine its power consumption depending on a set of activity measures. A number of scenarios are run in the hardware to stress each functional block. The method identifies six functional blocks (clock tree, instruction management unit, processing unit, internal memory, L1 data cache and L1 instruction cache). It does not consider what happens beyond the L1 cache. The measures considered include L1 hits, CPU stall cycles, number of executed instructions, etc. A non-linear model is obtained using curve fitting and the resulting equations that define the power for each functional unit depend on a single activity measure. For example the power of the processing unit depends only on the number of executed instructions.

2.1.2 System interconnect

Another component that has received significant attention is the system interconnect (bus or a network-on-chip). The analysis done in [19] indicates that external memory accesses are a major source of energy consumption in an embedded system. The work concludes that this is especially the case in multimedia platforms where power associated with off-chip accesses can dominate the overall power budget since the algorithms are more memory bound. Networks-on-chip are considered in [20]. The work presented in [20] indicates that power estimation considering only the total volume of data transported incurs inaccuracies since it abstracts away congestion. The proposed rate-based methodology measures traffic in a given sample period similarly as the approach follow in [20]. The rate-based methodology is used in [20] with linear regression to develop power models for the Hermes NoC where it is shown to decrease the error to around 6% compared with Synopsys PrimePower.

2.2 Power modeling at the system level

This section reviews power modeling that considers a system form by a set of interacting components.

2.2.1 Power modeling based on high-level activity

The proposed research falls into this category in which reference power consumption values are obtained based on a power estimation tool. A similar power estimation methodology targeted at the system-level is presented in [21]. The processor power model is not based on regression analysis but is simpler, based on just two states (busy and idle) with power assigned to each state based on an average obtained over many clock cycles. Linear

regression is used for the interconnect with a model of higher complexity than that for the CPU. The reference power data for each of these two models is based on pre-layout data obtained using a proprietary tool called CubicPower. The proposed work is based on post-layout data since our experience tells us that otherwise significant inaccuracies are introduced especially for modern processor cores such as the Cortex application processors that have customized implementation flows for optimal efficiency. The use of effective power management by the OS has been shown in [22] in which a power model allows the OS to obtain accurate estimates of a running process power consumption. The model is similar to the proposed approach although only based on four set of events for the four available performance counters in the selected computer system. Our model uses more low-level events and introduces the concept of states (e.g core active, core stall). Initial experiments showed that these additional measurements were needed in the proposed approach to obtain the target level of accuracy of around 5%.

This include measures taken from the performance counters available in the microprocessor cores but also other states and events manually selected in the SoC and not available to the software. A similar approach is used in [23] in which event counters are used to define an energy-aware scheduling policy which is implemented in the Linux Kernel. The approach is applied to a commercial architecture/board and it can only use the performance counters available to the software running in the microprocessor. In both cases energy modeling is used to better understand how applications are consuming the available energy and take actions affecting clock rate and voltage scaling to save energy. Our objective is different since we want to be able to make energy-aware system-level decisions at a pre-silicon stage when RTL code (or parts of it) is available but no system has been built.

2.2.2 Power modeling based on physical measurements

Physical measurements are generally considered the most accurate way of determining power consumption but they do have limitations. Firstly, physical measurements need fabricated chips which is too late and expensive in the design cycle. Secondly, process variability is expected to distort the measurements obtained from a limited number of fabricated chips.
System power analysis using physical measurements in a smartphone is performed in [5]. In this case physical power measurements are taken at the component level on a piece of real hardware. This is possible because the smartphone under consideration (Openmoko

FreeRunner) includes this capability but this is much more limited in general commercial devices. The device is instrumented to measure the power of CPU, RAM, GSM, LCD panel and touch screen, LCD backlight, WiFi, etc while running an Android 1.5 port and diverse applications. The backlight is found to be the main source of power consumption when the screen is on. The results also show that DRAM power can exceed CPU power in certain synthetic workloads but in practical situations CPU overshadows DRAM by a factor of two or more.

Instead of using the physical probes available in an experimental device to measure the power of individual components the work in [24] monitors overall power comsumption in a standard smartphone while running a set of applications that stress different parts of the system. The data obtained in this way is used to build the power models using linear regression techniques. The smartphone is then used normally by users while specially designed software logs the activity of the different parts of the system. The log data and power models are then used to identify the screen and the application processor as the main sources of power comsumption. An optimization is proposed that gradually reduces screen brightness to achieve a 10% reduction in total system power.

The following limitations have been identified in the previous works that are addressed in this paper. Firstly, power is calculated using pre-layout information so the accuracy will be reduced. In contrast our power models are derived from data obtained after a full implementation of the RTL designs and use PrimeTime PX [25] as the source of the reference power estimate. PrimeTime PX is a standard tool, and its level of accuracy is accepted by the industry, given properly characterized libraries and silicon process models. Secondly, the results tend to be based on benchmarks not necessarily representative of realistic use cases. Finally, modern multi-core configurations are not considered.

3. Methodology

This section presents the power characterization and modeling methology targeted to IP components that have RTL available that can be implemented before real silicon is available.

3.1 Methodology overview

The presented methodology provides a way of estimating the power or energy consumption of computer systems under real use cases with a reasonable simulation speed and computational complexity. In our systems of interest, a significant portion of the power

budget is spent in complex SoCs, which primarily are implemented using the standard cell based methodology. The industry standard way of accurate power analysis of these designs relies on RTL or post-layout netlist simulations. This process requires low level design information to be able to deliver accurate power estimates. The required inputs include a post-layout, clock tree inserted netlist, extracted parasitics, characterized libraries used at synthesis, and most importantly a very detailed (per net) capture of the switching activity of the design. We call this standard power analysis methodology "back-end power analysis" as it can be performed in the back-end ASIC design flow. There are two main reasons why back-end power analysis is slow and is infeasible for full application workloads. Firstly, capturing the switching activity required as an input is slow, because it requires RTL or netlist simulation and the monitoring of every net in the design. Secondly, the power estimation process itself is slow, for the same reason of having to account for every net in the design. We found that the application of regression modelling to estimate the power or energy consumption of RTL IP can provide sufficient accuracy within a 5% error, while maintaining the number of input variables low, therefore removing the major speed limiting factors mentioned above. Regression analysis can also be used to model other system components, which are not implemented using RTL synthesis.

The granularity of power estimation is at the component level, e.g.: CPU cores, slave components in a SoC or commercial of the shelf memory chips. It is useful to compose total power consumption as the sum of static power consumption and active power consumption:

$$P_{Total} = P_{Static} + P_{Active}$$

Static power is the part which is independent of the actual activity of the component, or the activity only has second order effect on this term. Active power is the part which is primarily determined by the activity of the component, but it can also depend highly on other factors. These two terms represent the leakage and dynamic powers of a SoC component, but the separation is also useful for describing other components of a system.

In this paper we present the methodology and results apply to active power modeling of RTL IP while its extension to include static power will be conducted in future work. The current research is based on a silicon process optimized for low leakage (TSMC 40LP), and therefore the static power represents a small percentage of total power and this has prioritized active over static power. Additionally, the higher threshold voltages needed to obtain low leakage means that the supply voltage in the low power process must be increased and the dynamic power is actually higher compared with a high performance process at the same frequency.

If RTL is not available power models can be extracted from data sheets or direct measurement in running silicon. Direct measurements make sense for IP blocks that are going to be reused so silicon is already available and in principle should be more accurate than RTL characterization. Nevertheless, direct measurements assume that the available silicon allows the measurement of power corresponding to the individual IP block and it could also introduce inaccuracies due to process variability if the measurements are done on a single device.

3.2 RTL characterization flow

To accurately model the active power or energy consumption of a SoC component, we use regression analysis, where the dependent variable is the expected active power or energy consumptions for the given building block, while the regressors are high level metrics of activity. The high level activity metrics used should be easy to capture from simulation. The choice of these metrics is a critical consideration in building power models for IP blocks and requires a detailed understanding of the operations of the component. Section 5 details the power models developed for the components considered in this work.

The other important requirement of regression analysis together with the choice of the regressors is the quality of the sample used for regression. To ensure high accuracy of our models, we use the industry standard back-end power analysis flow to produce our reference power or energy consumption data. Fig. 2 illustrates our RTL characterization flow, which involves the following steps:

1. Create a representative reference implementation of the RTL using the target process technology.
2. Create representative power benchmarks for the IP.
3. Create a test bench to simulate the execution of power benchmarks on the RTL.
4. Simulate a large set of power benchmarks on the test bench. Capture the high level activity metrics and the switching activity.
5. Using the back-end power analysis flow, extract the power or energy consumption of the power benchmarks based on the captured switching activity using the reference implementation.
6. Build the regression models based on the captured high level activity metrics, and corresponding power/energy consumption estimates.

There are some important thoughts that should be kept in mind about the steps.

Firstly, as the actual power consumption of the implemented design has a significant dependence on the actual implementation flow, the reference implementation ideally should match the target implementation. The reference implementation should be done using the same technology libraries, timing constraints, clock tree structure, process corner, floor plan, power/clock gating strategy, utilization factor, etc expected in the target design. In most cases, some of these settings must be approximated based on user experience because they are not known before the definition of the final chip.

When creating the power benchmarks, it is important that they are designed in such a way, that they exercise the whole of the relevant activity space, i.e.: the power benchmarks must make sure that all possible activities which significantly influence energy consumption are discovered during the characterization process.

Building the test bench for RTL simulation is conceptually straightforward. One can use either directly the RTL or the netlist obtained through the reference implementation flow. Netlist simulation is an order of magnitude slower than RTL simulation. The benefit of netlist simulations is that we can obtain more accurate power estimates from the back-end power analysis tools, as there is direct access to capture the complete set of low-level activity of the implementation. RTL simulation is faster, but we can only capture the switching activity of synthesis invariant nets, therefore the reference power estimates are less accurate. Based on our experiments, we found that RTL switching activity based power estimates still provide an acceptable level of accuracy for the reference data, within 5% difference from the values provided by netlist simulation. Therefore we choose this option to benefit from faster simulation speed for the characterization process.

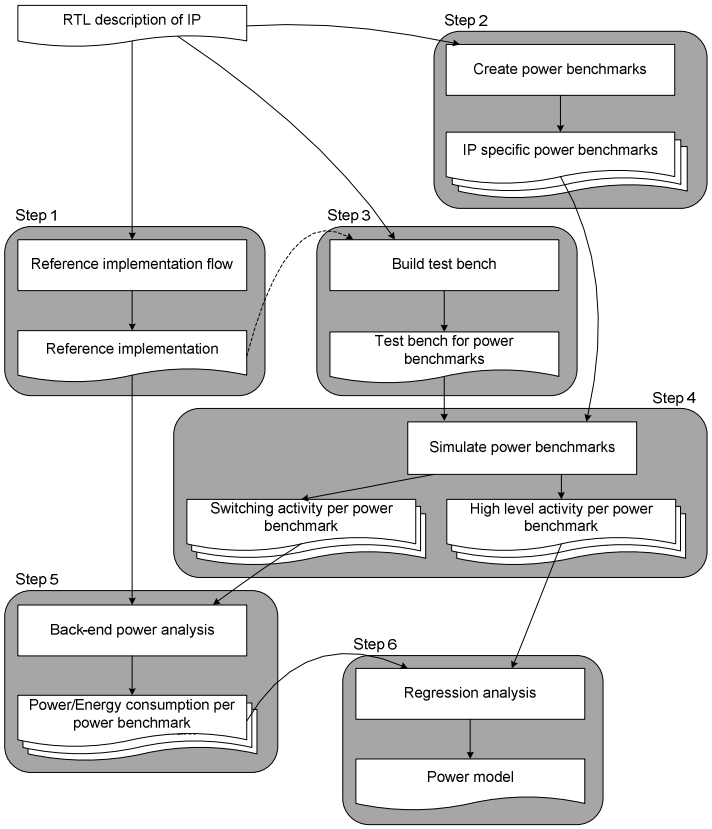

Fig. 2. RTL IP characterization flow

The simulations of the power benchmarks themselves are also straight forward. The low-level switching activity needs to be captured. This can be done using the standard VCD format or an aggregate switching activity format (e.g.: SAIF). For average power analysis in the back-end flow, both formats give the same result. VCD files tend to be very large in size and they grow with simulation time. We use the aggregate switching activity format, as we are only interested in total energy consumption (average power), and their fixed size makes them easier to handle. The high level activity metrics also need to be captured. There is no standard format for this. We simply use RTL counters connected to signals indicating the high level activities, and write the values of these counters into a text file at the end of the simulation.

Back-end power analysis is a standard step in the implementation flow, and provides an average power consumption value for the reference implementation, given an aggregate switching activity file. In our work, we used Synopsys PrimeTime PX.

The regression model can be built based on the corresponding high level activity metrics and back-end energy estimates. We found that simple linear models have a satisfactory accuracy

within a 5% error, given that we use appropriate regressors. Ideally the regressors should be activity metrics that have a clear impact on power and that are high-level enough so that simulations that do not involve RTL can also be used to collect them. Two primary types of activity metrics are considered:

- *State like metrics* which measure the time spent in some form of state. A few examples in case of a CPU are: time spent not executing instructions (caused for example by waiting for external memory accesses) or time spent actively processing.
- *Event like metrics* which count the number of occurrences of some form of event. Examples for a CPU: Number of L1 data cache hits or misses, number of instructions executed. In case of memory or network interfaces examples can be: number of bytes transferred, number of transactions completed.

These means that Energy consumption can be estimated as follows:

$$E = \sum_{j=1}^{J} P_j T_j + \sum_{k=1}^{K} E_k N_k = \sum_{l=1}^{J+K} m_l a_l = \mathbf{m}^T \mathbf{a}, \qquad (1)$$

where:

J = Number of state like activity metrics in the model

K = Number of event like activity metrics in the model

T_j = Time spent in state j (activity metric value)

P_j = Power contribution of state j

N_k = Number of occurrences of event k (activity metric value)

E_k = Energy cost of event k

a = High-level Activity vector: $(T_1\ T_2 \ldots T_J\ N_1\ N_2 \ldots\ N_K) = (a_1\ a_2 \ldots\ a_M)$

m = Vector of unknown power model coefficients:

$$(P_1\ P_2 \ldots P_J\ E_1\ E_2 \ldots E_K)^{\mathrm{T}} = (m_1\ m_2 \ldots\ m_M)^{\mathrm{T}}$$

The model coefficients **m** can be calculated using for example the standard Least-Squares method based on the (activity vector, back-end power estimate) pairs collected from simulation.

4. Target system

The target system is illustrated in Fig. 3. and corresponds to the shaded area of Fig. 1. Its central component is the Cortex-A9 multiprocessor core. The Cortex-A9 multiprocessor is an application processor introduced by ARM in 2007. The Cortex-A9 is designed around a dynamic length, multi-issue superscalar, out-of-order, speculating 8-stage pipeline. Additional processing includes a wide SIMD data processing engine called NEON. In this research a dual-core configuration is considered representative of current mobile phone architectures while frequency is set at 800 MHz achievable in the low-power process investigated. Extending the methodology to other technologies or system configurations (e.g four processor cores) is straight forward after collecting the proper activity measures and adjusting the power models.

The L1 instruction and data caches are both set at 32 KB as shown in Fig. 3. The cores are connected through a cache controller to a shared unified L2 cache. For this investigation we vary the size of the L2 cache between 128, 256, 512 or 1024 KB configurations. There are two 64-bit bus interfaces to the cache controller shared by the two cores. The L2 cache controller also has two 64-bit ports connecting it to the on-chip interconnect. The interconnect uses the AMBA AXI3 protocol to connect a configurable number of masters and slaves, and provides additional interfaces to AHB or APB compliant components as well. It provides support for multiple outstanding transactions and out-of-order transaction completion. The interfaces to the masters and slaves can be synchronous or asynchronous with variable data widths. In the current configuration the interconnect has been configured with multiple synchronous clock domains and a fully connected crossbar. The memory controller implements the interface to the LPDDR2 (Low Power DDR2) PHY using an industry standard DDR PHY interface (DFI) [26]. The clocks in the system are set to 800 MHz for the processors, L1 and L2 caches, 200 MHz for the interconnect and 400 MHz for the memory controller and LPDDR2 memory. The processor and interconnect speed are chosen as representatives of the frequency in the TSMC 40LP process selected in this work while the LPDDR2 is set to its normal operating frequency. The memory models are behavioral models for LPDDR2 memory chips developed by Micron [27]. The dual data rate feature means that two 32-bit words are read or written per clock cycle. This is the reason why the DFI interface doubles the data width compared with the memory. The memory controller clocks at 400 MHz with a data width of 64 bits while the interconnect clocks at 200 MHz and uses an interface of 128 bits to match the bandwidth of the memory controller. A

third master is added to this system in the form of a traffic generator. This traffic generator is designed to generate a configurable amount of traffic in the interconnect directed to the external memory. This additional master could represent a GPU or hardware accelerator as shown in Fig. 1 in a real system. The HLA (High Level Activity) monitor collects all the trace information during the run.

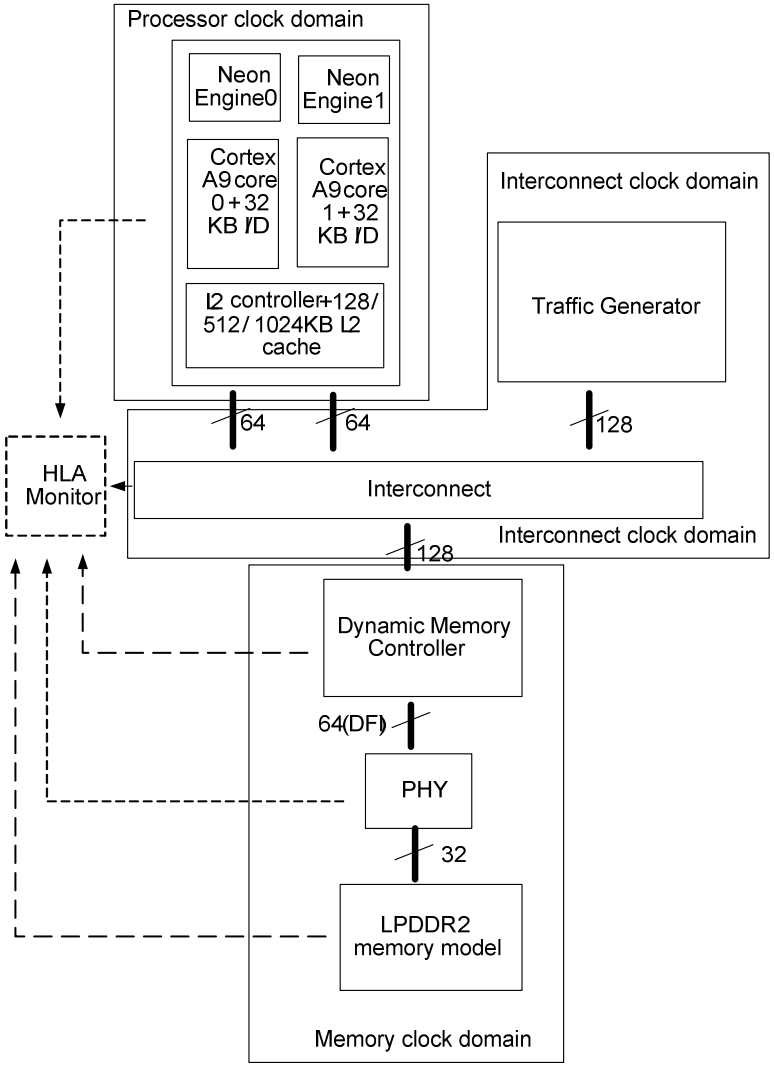

Fig. 3. Target system architecture

To simulate this system running realistic benchmarks and Linux in a traditional RTL simulator would need an unfeasible amount of time with a typical equivalent frequency of 100 Hz so we have used a specialized emulator machine (Cadence Palladium XP [28]). This emulator is a processor-based hardware/software verification and computing platform. RTL gets compiled into binary code that runs on the specialized processors available in the emulator. The designs run up to an equivalent frequency of 4 MHz. This is a fraction of the performance that could be obtained with a FPGA-based emulation but it has the advantage that the compilation runs are much faster than the place and routing jobs needed for the

FPGA. The high-level activity data is collected in the HLA monitor unit. Once this activity data has been extracted from the emulator the power models can be used to observe how system energy changes in each of the system components with time.

5. SoC power models

The linear regression methodology described in section 3 was used to obtain the power models for the memory controller, interconnect and processor cores which are IP cores developed by ARM and RTL is available for them. The power models for the LPDDR2, L2 cache and PHY were obtained based on data sheets and available Spice models. A summary of the activity metrics used for the power models in the system and the obtained normalized power coefficients are presented in Table 2.

5.1 CPU power model

To characterize the CPU power that includes the L1 caches, we built a test bench connecting the CPU to a DRAM controller via an AXI interconnect, in a similar setup as in our target system. The L1 cache is added to the CPU model because the fully implemented Cortex A9 hard macro already includes the L1 cache and removing it will imply the design of a new hard macro that will not represent the available product. The L1 is considered as an additional unit that forms part of the processor implementation and its activity is traced using the available performance counters since it is expected to have an impact on power. We used random instruction sequences as power benchmarks. The random instruction sequences have a constrained proportion of integer to floating point instruction ratios and contain typical program structures such as one and multi dimensional loops, memory intensive regions, etc. to ensure the activity space is thoroughly covered.

We found that the performance counters of the Cortex-A9 processor are good activity metrics for power modeling, and we mainly used these metrics as regressors. We also used a few additional terms, which are derived from the performance monitor signals through simple Boolean functions. Using the performance counters as regressors has the additional benefit that they can be captured easily from silicon, therefore validating the model against a test chip is relatively easy. Fig. 4 shows the histogram of model error compared to the reference power data.

5.2 L2 cache power model

The L2 memories are generated by a memory compiler as hard macros. To obtain a power model for the L2 we make use of the data sheets generated by the memory compiler. The data sheets provide information on the read and write currents and voltage settings so it is possible to estimate the power used by each type of access to each memory block.

The L2 cache memory is set associative with a total of 16 ways except for the smaller configuration of 128 KB with only 8 ways due to limitations in the memory compiler.

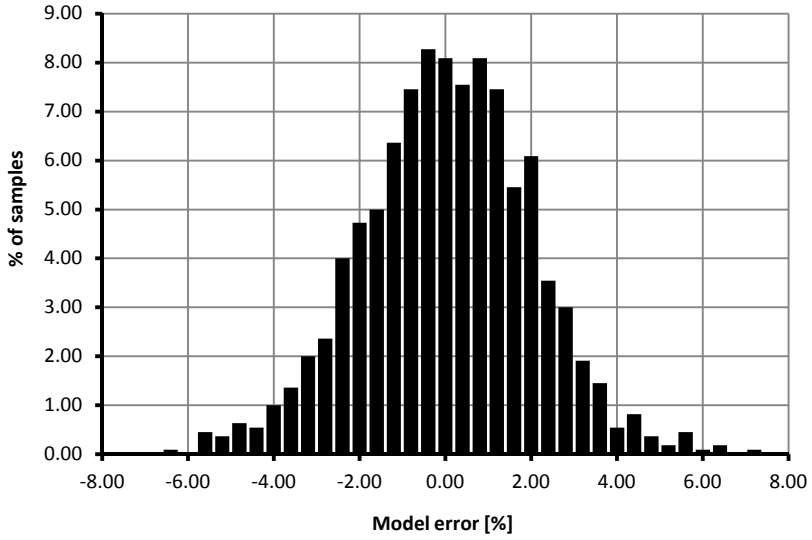

Fig. 4. CPU power model error

The architecture is shown in Fig. 5 for the 1 MB configuration with a total of 16 ways and 32 bytes per cache line. The RAM block SRAM2KX128 corresponds to a memory with 2K words and 128 bit per word and is generated with the Artisan high density single port SRAM RVT memory compiler using the TSMC 40nm LP libraries. Each RAM block contains 32 KB of memory and it is replicated 32 times to generate the 1 MB data memory. A tag memory word is associated to each L2 cache line. Similar memory blocks are generated for the other cache configurations considered in this investigation. The power implications of each type of access are summarized in Table 1. Table 2 shows the model coefficients for each of the L2 cache configurations : 1024, 512, 256 and 128 KB.

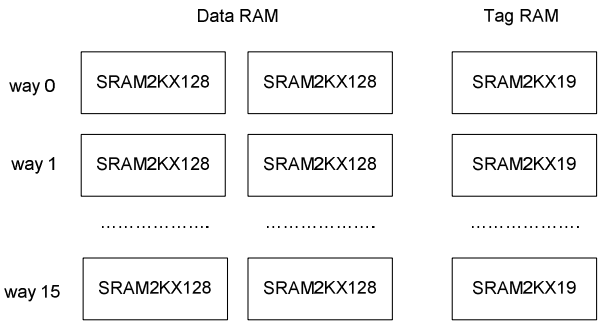

Fig. 5. L2 cache organization

Table 1. L2 power model components

	Read data power	Read tag power x16 (x8 for 128KB)	Write tag power	Write data power
Data read hit	yes	yes	no	no
Data read miss	no	yes	yes	yes
Data write hit	yes	no	yes	yes
Data write miss		yes	yes	yes
Instruction read hit	yes	yes		
Instruction read miss		yes	yes	yes
Data write miss with eviction	yes	yes	yes	yes
Data read miss with eviction	yes	yes	yes	Yes

5.3 memory controller power model

Similarly to the CPU, we used randomized stimulus as power benchmarks to characterize the memory controller power model. We connected the memory controller to a traffic generator which generated randomized read and write requests, with different constraints on total bandwidth requested, proportion of read and write operations, transaction burst lengths, inter transaction time distributions, etc. Fig. 6 shows the histogram, of the model error.

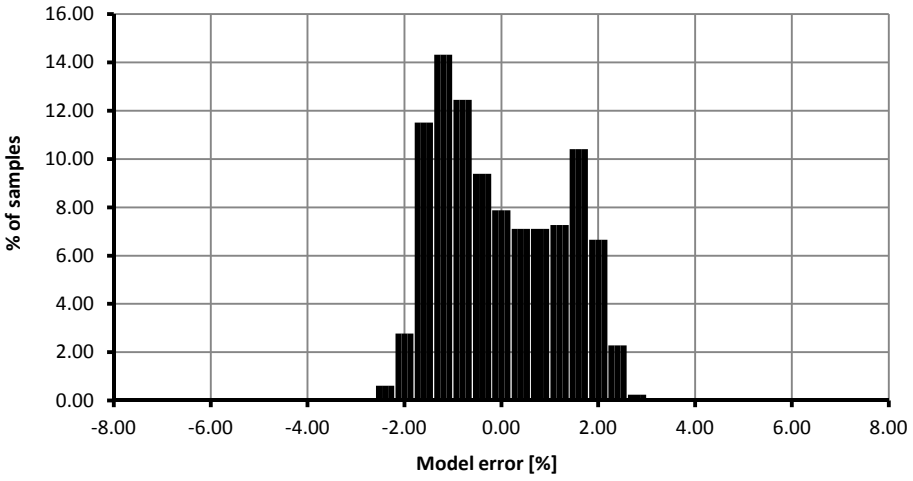

Fig. 6. Memory controller power model error.

5.4 Interconnect power model

To develop the power model for the interconnect the CPU cores in Fig 3 were replaced by two traffic generators capable of sweeping the activity space generating different traffic types. The third traffic generator shown in Fig 4 represented a second master type (e.g GPU, hardware accelerator). There are a two main reasons why the CPU RTL models were not used directly. Firstly, power characterization should be based on benchmarks that exercise the interconnect thoroughly sweeping the activity space with different types and volume of traffic and this is easier to control with simple traffic generators. Secondly, adding the RTL simulation of the Cortex A9 multiprocessor will slow down the simulation unnecessarily. The model error that measures how the resulting linear equation estimates power against the PrimeTime PX results is shown in Fig. 7.

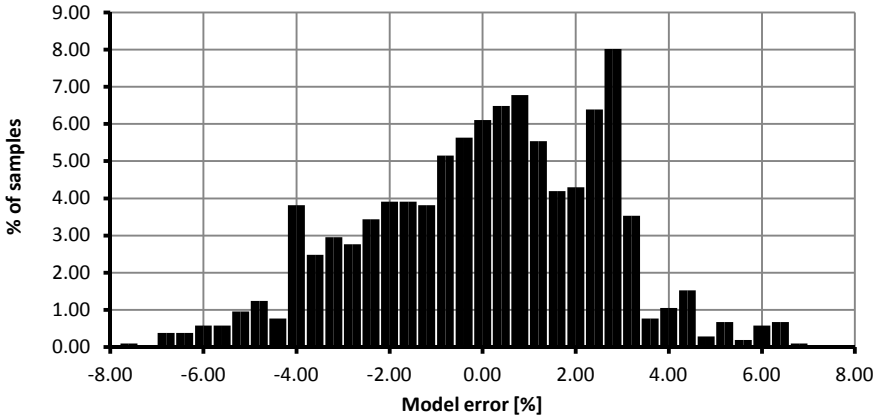

Fig. 7. Interconnect power model error.

5.5 LPDDR2 and PHY power models

The power model for the LPDDR2 memory device is built based on the methodology described in [29], adapted for LPDDR2 memories, using data from publicly available datasheets from [27]. The PHY is characterized based on SPICE simulation at different bandwidth levels.

Table 2. Power models activity counts.

	Model Summary		
Parameter	**Value**	**Type**	**Description**
core_state_active	100		Counts the number of cycles the core is in active state executing instructions
core_state_stall	112		Counts the number of

		CPU state	cycles the core is in stall state waiting for some additional inputs before progressing
core_state_wfi	0.12		Counts the number of cycles the core is in standby mode with most of the clocks disable.
intc_state_clock_enabled	82		Counts the number of cycles during which the integer clock is enabled
neon_state_clock_enabled	51		Counts the number of cycles the neon data engine unit is enabled
Integer_instruction_renaming	0.11	CPU event	Number of instructions going through the register renaming stage.
floating_point_instrucion_renaming	0.19		Counts the number of floating point instructions going through the register rename stage
neon_instruction_renaming	0.05		Counts the number of neon instructions going through the register rename stage
d_cache_miss	0.87		Counts the number of L1 data cache accesses that resulted in a data cache miss
i_cache_miss	0.22		Counts the number of L1 instruction cache misses.
data_read_hit	101.0/93.5/90.5/57.2	L2 cache event	Counts the number of L2 data cache read accesses that result in a cache hit.
data_read_request	131.2/121.0/116.7/83.89		Counts the number of L2 data cache read accesses.
data_write_hit	103.9/96.2/92.8/59.52		Counts the number of L2 data cache write accesses that result in cache hit.
data_write_request	131.2/121.0/116.7/83.89		Counts the number of L2 data cache write accesses.
Instruction_read_hit	101.0/93.5/90.5/57.20		Counts the number of L2 instruction read accesses that result in a hit.
Instruction_read_request	131.2/121.0/116.7/83.89		Counts the number of L2 instruction read accesses.
write_channel_cpu_0	0.065		Counts the number of write transfer in the cpu0/axi interface
read_channel_cpu_0	0.075		Counts the number of

		Interconnect event	read transfer in the cpu0/axi interface
write_channel_cpu_1	0.073		Counts the number of write transfer in the cpu1/axi interface
read_channel_cpu_1	0.078		Counts the number of read transfer in the cpu1/axi interface
write_channel_tg	0.055		Counts the number of write transfer in the tg/axi interface
read_channel_tg	0.043		Counts the number of read transfer in the cpu/axi interface
read_channel_slave	0.055	Memory controller event	Counts the number of read transfer in the memory controller/axi interface
write_channel_slave	0.056		Counts the number of write transfer in the memory controller/axi interface
dfi_write_data_enable	112.5	PHY event	Counts the number of write enables in the dfi/memory chips interface.
dfi_read_data_enable	112.5		Counts the number of read enables in the dfi/memory chips interface.
activate_command	5.98	LPDDR2 event	Counts the number of activate commands in the memory chips.
read_command	2.16		Counts the number of read commands in the memory chips.
write_command	1.83		Counts the number of write commands in the memory chips.
clock_enable_all_banks_precharge	24.97	LPDDR2 state	Counts the number of clock enable with all banks precharched events the memory chips.
clock_disable_all_banks_precharge	0.85		Counts the number of clock disable with all banks precharched events the memory chips.
clock_enable_some_banks_precharge	30.46		Counts the number of clock enable with some banks precharched events the memory chips.

5.6 System power model validation

In principle, the ideal way to validate the system power model is to measure silicon implementing the IP components and then compare these values with the values obtained from the power model running a set of realistic applications. As already mentioned this method is not perfect since it does suffer from inaccuracies from the variability present in deep-submicron implementations and measures taken in a single or small set of boards. In this research such a real-silicon implementation was not available so an alternative approach was used. A set of applications was selected as a validation data set independent of the applications that had been used to characterize the models. A simulation of the RTL was performed running these applications and both the high level activity metrics and low level switching activity were collected. The back-end power analysis flow was used to extract the power and energy consumption of the validation benchmarks and the obtained results were compared with using the power models and the high level activity metrics. The results are shown in Fig. 8 as a relative error. Notice that this error must be added to the error introduced by back-end power analysis flow to obtain an absolute error measurement. Synthetic benchmarks such as CPU stress show a very high accuracy with more realistic benchmarks such as matrix multiplication and FFT showing around a 5% error.

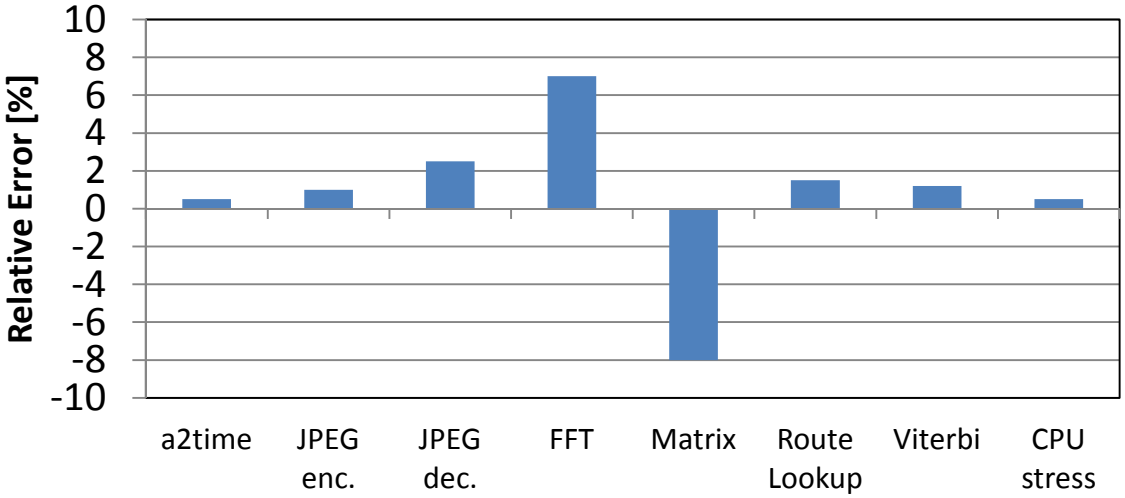

Fig. 8. Power model validation.

6. Power and Energy analysis

This section uses the power models developed in section 5 to analyse the power and energy consumption for a set of realistic benchmarks representing web browser activity and video coding. Additionally, we also select a number of applications extracted from the SPEC CPU2000 benchmarks. The experiments highlight the effects of non-CPU components in system power. For reasons of market confidentiality the figures have been normalized and the units in the axis removed. In any case, the contribution of the paper is not to show these absolute values but the relative changes in energy and power depending on system configuration and executed benchmark.

6.1 Benchmarks description

The browser activity benchmark is called *Bbench* [30]. *Bbench* is a set of browser benchmarks that can be used to measure web page render performance of a browser on a target system over a number of pages. In the configuration used in this work five pages are used corresponding to content from Amazon, ESPN, Wikipedia, Google and Craigslist. The browser selected is Firefox 3.5 and the benchmark performs a total of five loops loading and rendering all five pages in each loop. The first loop starts with empty caches while the idle time between each page load is 1 ms. This gives enough time to the browser to finish rendering the page before moving to the next one. We also select a proprietary implementation of the popular H.264 video codec to explore the power and energy consumption of a typical multimedia application. This proprietary implementation corresponds to the decoder part and it is multi-threaded based on OpenMax [31] libraries. The decoder is configured with four parallel threads. Neither the SPEC CPU2000 benchmarks nor the Firefox browser are multi-threaded but the Linux scheduler can launch them concurrently so that both cores can be active simultaneously. The SPEC CPU2000 benchmarks used are vpr: (FPGA circuit placement and routing), gzip (Lempel-Ziv based lossless compression algorithm), eon (probabilistic ray tracer), bzip (Burrows-Wheeler transform compression algorithm with arithmetic coding), mcf (Combinatorial optimization: single-depot vehicle scheduling), crafty (chess game playing) and twolf (standard cell place and route). All the benchmarks run under Linux kernel version 2.6.28 and the compiler used is arm-linux-gcc-4.3.2.

6.2 SPEC CPU2000 and video codec power analysis

Fig. 9 shows the power analysis for the set of SPEC CPU2000 benchmarks and the H.264 codec. Linux boot completes after the first 4 seconds of execution and the benchmarks are run sequentially. The power associated with L2 cache, PHY, memory controller and interconnect have been grouped in a non-core component to simplify the graph.

Fig. 9 shows that the main source of power consumption for these benchmarks is the CPU but the combination of non-core and memory is comparable for some runs such as *gzip2, bzip1 and bzip2* indicating that power optimization based only on CPU data as done in the processor focused techniques reviewed in section 2 could be misleading. Fig. 9 also shows that power consumption varies significantly with the application and also with the data processed during each run. The first two algorithms *vpr* and *twolf* consume similar power but this is considerably different from the power used by the *eon1* and *eon2* benchmarks. This could be due to the type of instructions used by the eon benchmarks being more power intensive than the *vpr/twolf* benchmarks or different instruction level parallelism so that more execution units are active in parallel. The power used in these four benchmarks in the memory and non-core components is comparable which is confirmed by observing the cache miss rates indicating that the access rate to lower levels of the memory subsystem is similar.

Significant differences can be found between the *gzip1* and *gzip2* runs. The input data in the second case is more randomized and more difficult to compress. The algorithm generates a larger amount of accesses to the lower levels of the memory subsystem which is reflected in the higher power in the non-core and LPDDR2 components. The total power used by *gzip2* is clearly higher than *gzip1* but, on the contrary, *gzip1* CPU power is comparable to *gzip2*. This indicates that power optimization should focus on the system and memory hierarchy since in this case *gzip2* is a more power intensive benchmark.

The two *bzip* runs compress the same data as for *gzip*. In this case the randomized data does not need significantly more power but it increases the execution run time considerably, therefore increasing the energy needed by the application. LPDDR2 power is higher than CPU during some time intervals and the non-core plus LPDDR2 comparable to the CPU power again.

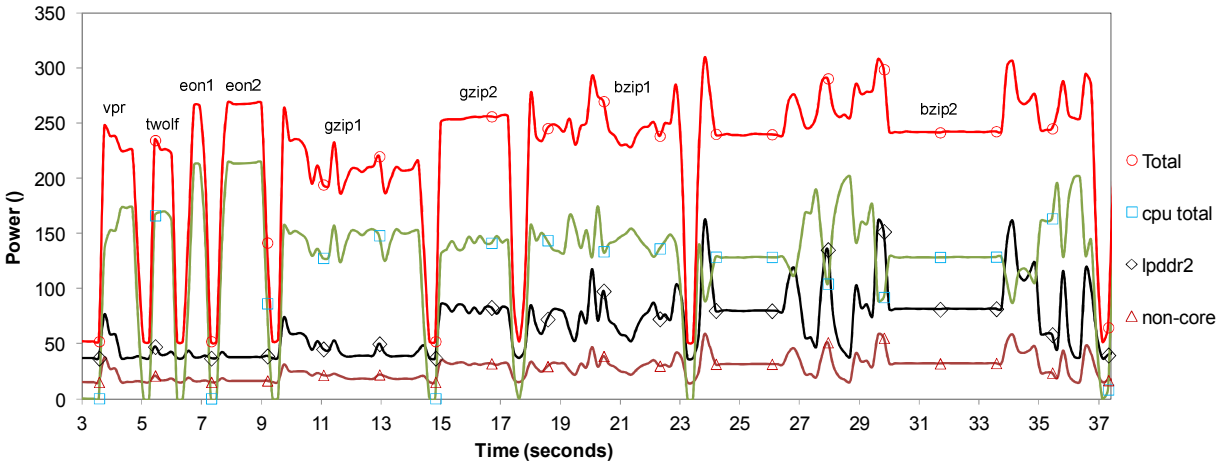

Fig. 9. Power analysis of the system components for the SPEC CPU2000 benchmarks (normalized power in Y axis).

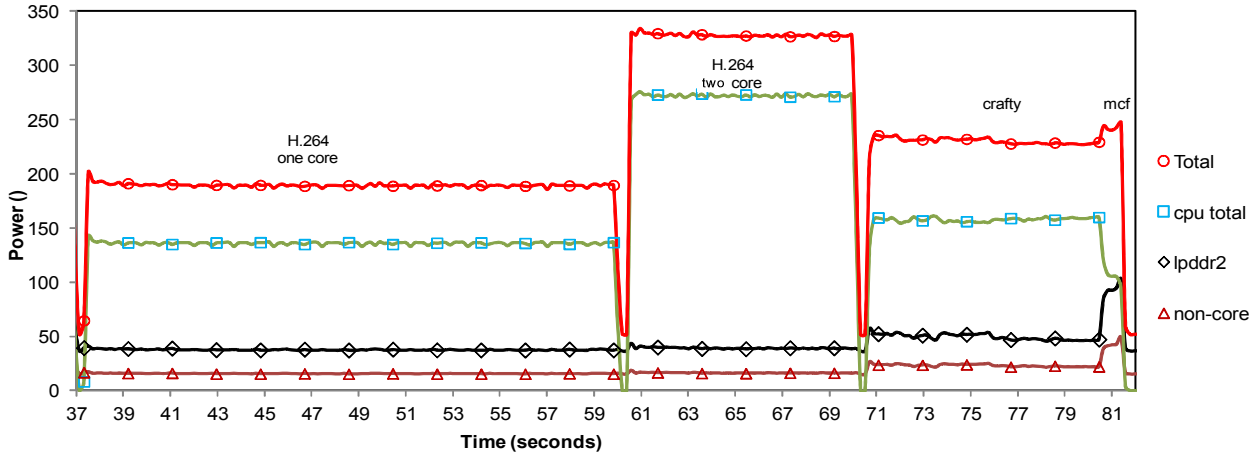

Fig. 10. Power analysis of the system components for the SPEC CPU2000/H.264 benchmarks (normalized power in Y axis).

Fig. 10 shows the results for the *H.264* runs and two additional SPEC CPU2000 benchmarks. Both *H.264* runs decode the same number of VGA frames from the same sequence but in the first case only one processor core is active while in the second case both processor cores are active so the 4 parallel threads can be distributed by the OS. The dual-core configuration execution time is approximately 10.2 seconds while for the single core is 23 seconds. This indicates a speed up factor of 2.25. This super-linear speed up can be explained with the effects of utilizing two L1 caches in the dual-core configuration, increasing the effective cache size. This means that thanks to L1 cache snooping a core can find data in the second cache which is not present in its own cache reducing accesses to the L2 cache.

The *mcf* benchmark is a highly memory bound application and this is reflected in the results. The power used by the memory subsystem and non-core components is clearly larger than the processor core. It is also interesting that the power of the CPU is lower compared to *crafty* but this is not necessarily due to the instruction mix but the fact that the processor stalls more waiting for the memory subsystem to supply the required data. The resulting effect is that the power used by *mcf* is higher than *crafty* although the CPU power itself is lower highlighting once more the importance of system optimization.

The results observed in these experiments can be used to determine the maximum power needs and the changes of power with time in the system dependent on data inputs and application reducing the requirements for over pessimistic design of components such as voltage regulators, board design, etc. The experiments also clearly indicate that non-core and system memory power can be higher than CPU power for realistic applications. This means that architectural decisions that lead to a reduction in CPU power could lead to an overall increase of power consumption if the power of system components increases. Therefore to be useful and accurate power modeling techniques need to consider system effects. In the following section we illustrate this point with an example that shows how an architectural decision that seems to reduce CPU power increases overall system power and also system energy consumption.

6.3 Web browsing power analysis

Fig. 11 shows the power results from running the browser benchmark for different L2 cache configurations. The first noticeable effect is that total power increases as the L2 cache size decreases (total_l21024k to total_l2128k), since more accesses are passed to the external memory. On the other hand, Fig. 11 also shows that CPU power decreases for the smaller L2 configurations (cpu_l21024k to cpu_l2128k). The reason for this is that the smaller L2 caches cause more stalling in the CPU while waiting for data to arrive from memory and this means that the power is lower since the CPU is idle. By observing the peaks in CPU power it is possible to identify when the browser loops finish as indicated in Fig. 11 (cold loop and warm loops). The cold loop starts with empty caches while for the other four loops the caches have been filled with data and instructions (warm loops). Memory power increases during the warm loops for the smaller cache configurations (lpddr2_128k and lpddr2_256k). It is basically constant for the 512K configuration and decreases slightly for the 1024K cache configuration. The reason is that the larger cache configurations reduce the traffic to external

memory but for the smaller configurations additional traffic takes place with data being evicted from the L2 cache during the warm loops.

Fig.12 shows the average power for the components considered in Fig. 11 as a function of the L2 cache size. The figure shows how the average power of the CPU increases slightly as cache size increases while the opposite is clearly true for the system power. Web page rendering is a task with clearly identifiable on and off periods in contrast with long-running applications such as video decoding. Energy optimization can be seen, therefore, as a more appropriate objective instead of power optimization. Fig. 13 compares the energy requirements of the browser benchmarks for the different L2 cache configurations. Total energy includes everything (multiprocessor CPU, L2 caches, memory and the rest of the components) and typically doubles the CPU only measure especially for the smaller cache configurations. The energy corresponding to the CPU shows that the reduction obtained from using larger L2 caches is small. The effect here is that the run time of the benchmark with the smaller cache increases and this should increase energy considerably but the power of the CPU with the smaller L2 caches decreases as seen earlier. Both effects (larger execution time and lower average power) cancel each other to some extent. On the other hand, looking at total energy the positive effect of the larger caches is more significant with an energy reduction of 34 %. This indicates that the system must be considered and CPU only could lead to wrong conclusions like that the L2 cache size has a relatively small effect on energy.

6.4 Single and dual core comparison.

Fig. 14 and 15 show the power and energy results comparing the execution of the benchmarks considered in section 6.2. In this case the Linux scheduler receives all the benchmarks in parallel and decides when each benchmark runs or switches out. If the benchmarks are run sequentially as seen in the section 6.2 experiment the scheduler cannot do anything but wait when the current task is not ready (e.g I/O) since they are all single-threaded with the exception of the parallel H.264. In this experiment, however, the scheduler can assign a different benchmark (as a separate process) to execute instead of stalling. This is true for both the single and dual core configurations. The single core configuration completes the whole set of benchmarks after 62.8 seconds while the dual core completes after 36.5 seconds obtaining a 72% speed up.

The energy comparison in Fig. 14 shows that the energy used by the dual core configuration is slightly larger than the single core (3% higher) while the energy used by the LPDDR2 and the non-core are lower for the dual core (19% and 20% respectively). Total energy is also

lower for the dual core (5% lower) indicating the importance of considering the system from an energy point of view. It is important to note that our test system does not implement dynamic voltage and frequency scaling. If this was the case then the dual core configuration could have scaled down its voltage/frequency operating point to complete the benchmarks in the same amount of time as the single core. It is expected that this will result in a lower dynamic energy consumption for the dual-core although doubling the amount of logic will also increase static energy. Further work should aim at investigating the different static/dynamic energy trade-offs in parallel implementations.

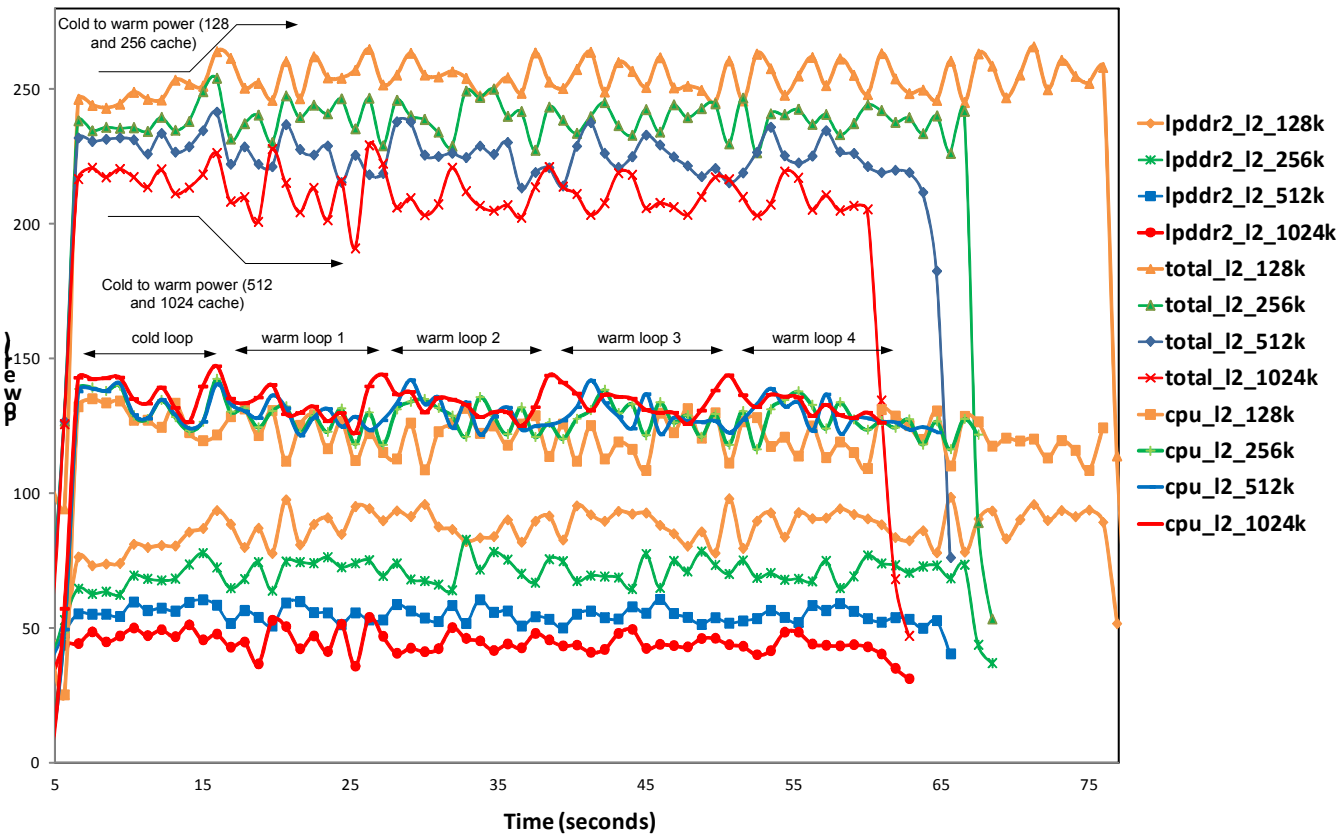

Fig. 12. Average power analysis for the browser benchmarks in function of the L2 cache configuration (normalized power in Y axis)..

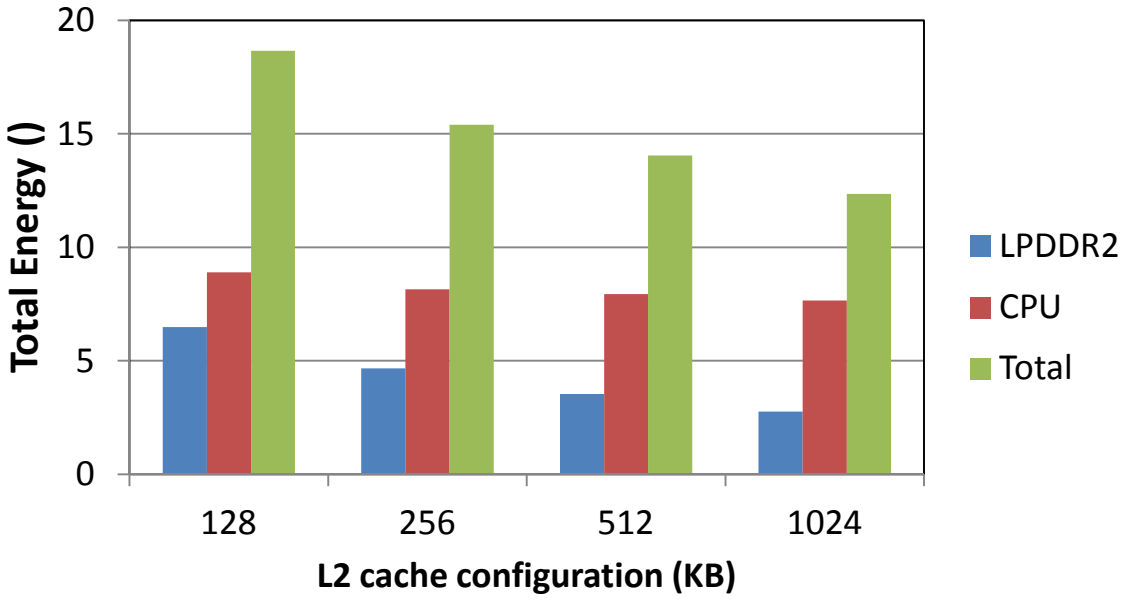

Fig. 13. Total energy analysis for the browser benchmarks in function of the L2 cache configuration (normalized power in Y axis).

7. Conclusions.

This research has presented a power modeling methodology based on power models developed using post-layout data and regression analysis. The usage of post-layout data enables a level of accuracy not possible with previous approaches that do not consider the implementation effects. The extension of the methodology to include the rest of the components in the system requires the addition of power models based on data sheet information or physical measurement for third-party components or components with no RTL description. The investigation is based on a realistic scenario in which a set of standard benchmarks are run after booting Linux in a multiprocessor hardware. The results highlight the importance of considering the whole system from a power point of view showing how changes in the system architecture can reduce CPU energy but increase system energy or increase average CPU power but reduce average system power. The experimentation shows that an energy reduction of 34% can be obtained replacing a 128KB L2 cache with a similarly configured 1024KB L2 cache during Firefox web browsing although the effects on CPU

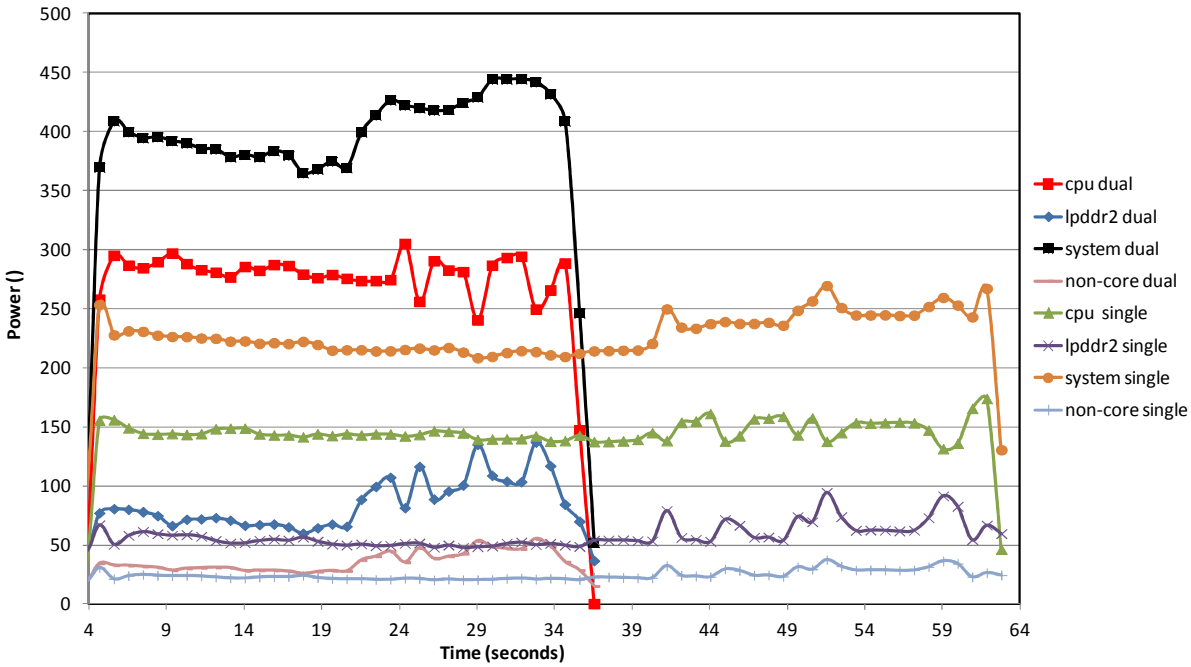

Fig .14. Power analysis for the SPEC CPU2000/H.264 benchmarks in function of core count.

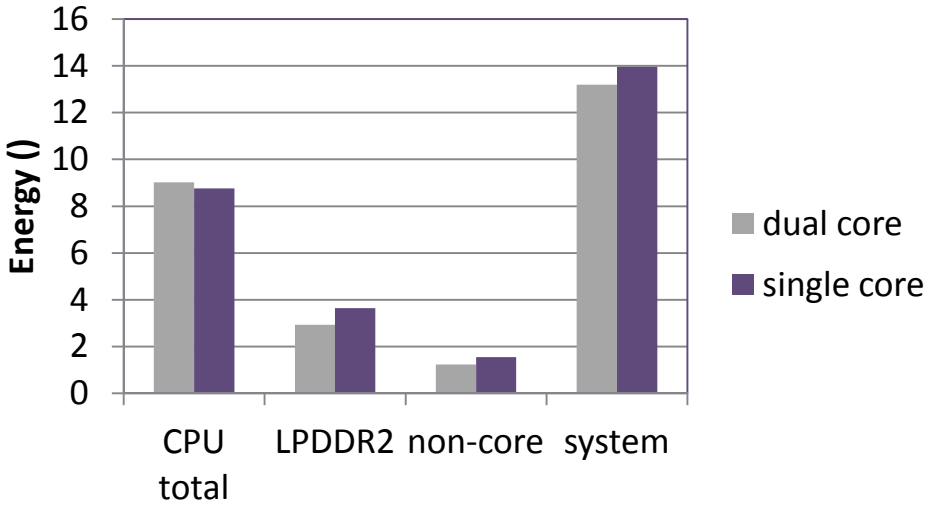

Fig. 15. Total energy analysis for the SPEC CPU2000/H.264 benchmarks in function of core count.

energy are much more modest. As future work we plan to investigate how the power models can be used in a high-level simulation environment to speed up power estimations. Additional refinements of the methodology will consider power control techniques such as dynamic voltage frequency scaling and also its extension to include static power. The power and energy results have been initially validated against applying the same methodology to an independent validation application set but an experimental board is also under development

that will enable direct power measurements of the system-on-chip and memory subsystem for further verification and correlation with the estimated results.

Dr Jose Nunez-Yanez is a senior lecturer in the school of Engineering at Bristol University, UK. During 2011 he was a research fellow at ARM Ltd with a Royal Society fellowship. His research interests are in the area of reconfigurable computing, energy efficient design and data compression.

Geza Lore got his M.Sc. in Electrical Engineering from the Budapest University of Technology and Economics in 2009. He is currently working as an engineer in the Research Group at ARM. His research interests are in energy-efficient microprocessor architectures and high-level system modelling.

1. S. Borkar and A. Chien. The future of microprocessors. Commun. ACM 54, Vol. 5 pp. 67-77, May, 2011.
2. Nam Sung Kim et al. Microarchitectural power modeling techniques for deep sub-micron microprocessors. In Proceedings of the 2004 international symposium on Low power electronics and design (ISLPED '04). ACM, New York, NY, USA, pp. 212-21, 2004.
3. C. H. van Berkel.. Multi-core for mobile phones. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '09), Belgium, pp. 1260-1265, 2009.
4. M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi and C. Turchetti. System-Level Power Analysis Methodology Applied to the AMBA AHB Bus. In Proceedings of the conference on Design, Automation and Test in Europe: Designers' Forum - Volume 2 (DATE '03), vol. 2, IEEE Computer Society, Washington, DC, USA, 2003.
5. A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In Proceedings of the 2010 USENIX annual technical conference (USENIXATC'10). USENIX Association, Berkeley, CA, USA, pp. 21-21, 2010.

6. N. Sung Kim, T. Austin, T. Mudge, and D. Krunwald. Challenges for architectural level power modeling. In Power aware computing**,** Robert Graybill and Rami Melhem (Eds.). Kluwer Academic Publishers, Norwell, MA, USA, pp. 317-337, 2002.
7. D. Burger and T. Austin. The SimpleScalar tool set, version 2.0. SIGARCH Comput. Archit. News 25, vol. 3, pp, 3-25, 1997.
8. D. Brooks, R. Dick, R. Joseph and L. Shang, Power, Thermal, and Reliability Modeling in Nanometer-Scale Microprocessors, Micro, IEEE , vol.27, no.3, pp.49-62, 2007.
9. D. Brooks, V. Tiwari, and M. Martonosi.. Wattch: a framework for architectural-level power analysis and optimizations. In Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00). ACM, New York, NY, USA, 2000.
10. N. Vijaykrishnan, M. Kandemir, M.J. Irwin, H.S. Kim and W. Ye, Energy-driven integrated hardware-software optimizations using SimplePower, Computer Architecture, Proceedings of the 27th International Symposium on , vol., no., pp.95-106, 14-14 June, 2000.
11. D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma and M. Rosenfield. New Methodology for Early-stage, Microarchitecture-level Power-performance Analysis of Microprocessors, IBM Journal of Research and Development, Vo. 47, No. 5/6, Nov. 2003.
12. Information available at http://www.arm.com/products/processors/cortex-a/cortex-a9.php (accessed 12/12/2012)
13. H. Jacobson, A. Buyuktosunoglu, P. Bose, E. Acar and R. Eickemeyer. Abstraction and Microarchitecture Scaling in Early-Stage Power Modeling. International Symposium on High-Performance Computer Architecture (HPCA), February, 2011.
14. B. Lee and D. Brooks. Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction, International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XII), San Jose, CA, October, 2006
15. M. Moudgill, J. Wellman, and J. Moreno. Environment for PowerPC Microarchitecture Exploration. IEEE Micro, Vol. 19, No. 3, pp. 15-25,May, 1999.
16. A. Kleinosowski, J. Flynn, N. Meares and D. Lilja. Adapting the SPEC 2000 benchmark suite for simulation-based computer architecture research. In Workload characterization of emerging computer applications. Kluwer International Series In Engineering And Computer Science Series, Vol. 610. Kluwer Academic Publishers, Norwell, MA, USA, pp. 83-100, 2001.
17. S. Nikolaidis et al. Instruction level energy modeling for pipelined processors. J. Embedded Comput. Vol. 1, No. 3, pp. 317-324, 2005.
18. M. Ibrahim et. al. A precise high-level power consumption model for embedded systems software. EURASIP J. Embedded Syst. Article 1, January, 2011.
19. K. Ning and D. Kaeli. Power Aware External Bus Arbitration for System-on-a-Chip Embedded Systems. In Transactions on High-Performance Embedded Architectures and Compilers I**,** Per Stenstr, Lecture Notes In Computer Science, Vol. 4050. Springer-Verlag, Berlin, Heidelberg. 2007
20. L. Ost et. al.. A high abstraction, high accuracy power estimation model for networks-on-chip. In Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes (SBCCI '09). ACM, New York, NY, USA, , Article 31 , 2009.

21. I. Lee et al. PowerViP: SoC power estimation framework at transaction level, Design Automation, Asia and South Pacific Conference on , No. 8, pp. 24-27 January, 2006.
22. D. C. Snowdon, E. Le Sueur, S. M. Petters, G. Heiser. Koala: a platform for OS-level power management. EuroSys: pp. 289-302, 2009.
23. A. Weissel and F. Bellosa. Process cruise control: event-driven clock scaling for dynamic power management. In Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems CASES'02, October, 2002.
24. A. Shye et al. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42). ACM, New York, NY, USA, pp. 168-178, 2009.
25. K. Kannan, D. Galbi, Measuring Active Power using PrimeTime PX -- A User Perspective Synopsys Users Group Conference, Boston, MA September 21, 2010,
26. Information available at: http://www.ddr-phy.org. (last accessed 12/12/2012)
27. Information available at: http://www.micron.com/products/dram/mobile-lpdram (last accessed 12/12/2012)
28. Information available at: http://www.cadence.com/rl/Resources/technical_briefs/palladium_xp_tb.pdf (last accessed 12/12/2012)
29. Technical Note - Calculating Memory System Power for DDR3, Micron http://download.micron.com/pdf/technotes/ddr3/TN41_01DDR3%20Power.pdf (last accessed 12/12/2012)
30. Anthony Gutierrez, et. al. , Full-System Analysis and Characterization of Interactive Smartphone Applications, accepted for publication in IISWC2011.
31. A. Gutierrez, R. Dreslinski, T. Wenisch, T. Mudge, A. Saidi, C. Emmons and N. Paver. Full-system analysis and characterization of interactive smartphone applications. In Proceedings of the 2011 IEEE International Symposium on Workload Characterization (IISWC '11). IEEE Computer Society, Washington, DC, USA, pp. 81-90, 2011.
32. Information available at http://www.khronos.org/openmax/ (last accessed 12/12/2012)