New Approach on Power Efficiency of a RISC Processor

OREST OLTU[2], VALENTIN VOICULESCU[1], GARY GIBSON[1],
LUCIAN MILEA[2], ADRIAN BARBILIAN[3]
[1]VIRTUALMETRIX DESIGN S.R.L.
Bd. Dorobantilor 168, Sect 1, Bucharest, ROMANIA,

[2] Electronics, Telecommunications and Information Technology Faculty,
Politehnica University of Bucharest, Iuliu Maniu 1-3, Bucharest, ROMANIA,

[3] Clinical Central Military Hospital "Carol Davila", Bucharest, ROMANIA,

Abstract: - Most often, for a digital electronic device, the consumed power is indicated as an average value without explicitly expressing the functional conditions under which this power was measured. This is also true for microprocessors. For a large part of the ARM (Advanced RISC Microprocessors) used in battery-supplied devices the consumed power represents a critical parameter. Any drop in power absorption from the battery is important. For designers working on improving this parameter, it is very useful to be able to have a different way to characterize the processor from an energetic point of view instead of the classical model of power consumption figures measured in microwatts per Megahertz. In this paper we propose a new method to compare two hardware variants of ARM processors, designed to work for a specific set of applications. Because the absorbed power varies dynamically during the time the application is running, we consider that the total energy required during this time for the run is a more exact and correct way to characterize the energetic efficiency of the microprocessor.

Key-Words: - RISC processors, energetic efficiency, power consumption, leakage currents

1 Introduction

A large part of the ARM family of processors is used to manufacture portable devices powered by batteries. This is the case for most of the consumer electronics products like mobile phones, PDA's, handheld media players and gaming devices, for which the time between two successive battery charges is an essential parameter for competitive reasons.

ARM Limited owns the IP's for these processors, but they are manufactured under license by an extensively large number of semiconductor companies, including Atmel and Infineon. The owner allows a client to customize its processor for a specific application, while keeping the core and usually the memory model and some of the peripherals of a standard part. This policy led to the widespread use of ARM processors in a very large number of products.

In the customization process for a specific application, each designer desires to obtain the most accurate energetic efficiency values to be able to compare between different hardware and software implementations.

In this paper, we propose a way to characterize the processor from this point of view, which we consider to be more precise than the usual one.

2 Problem Formulation

For starters, we have to remind the reader of a very simple thing that usually gets overlooked: power represents energy consumed in a time unit, conventionally in one second. For electronic devices power is usually obtained by measuring the current, the voltage being usually regulated. Most of the time we measure the continuous current absorbed from the power supply, and the value obtained for power is primarily used for device thermal design. The thermal processes vary slowly, with large time constants, and the quick variations of the absorbed current during runtime are implicitly integrated, so they don't matter from a thermal point of view.

For devices like the microprocessors, this current can change quickly due to dynamic conditions, even during the runtime of one given application.

The averaging done only by the measuring system can lead to a significant error for the obtained current value.

An exact measurement must take into account the current variation diagram on the entire runtime of a given application, and the computation required, must be done based on the integral given in (1):

$$\int_0^T i(t)\,dt = W(T) \qquad (1)$$

Where T represents the length of the application runtime, W(T) represents the energy consumed by the processor in the time interval T.
But it is possible, and quite common, that for two different implementations of a processor the application runtime to be different.
In this case, comparing the two implementations using as criteria the dissipated power is not adequate if we have in mind the battery consumption.
The correct comparison can be done only between $W_1(T_1)$ and $W_2(T_2)$, so among consumed energies, which are a measure of the electrical charge absorbed from the battery.
In this paper we present the results of measurements that confirm the existence of significant variations in absorbed current depending on the workload of an ARM processor. This workload can change repeatedly and during application runtime if a power management system is used, in order to reduce battery consumption.

3 Energy Measurement Method

We used in out study a development processor belonging to the ARM11 family, which has more memory and more peripherals than a standard ARM11 processor.
In Fig.1 we present the block diagram for this chip.

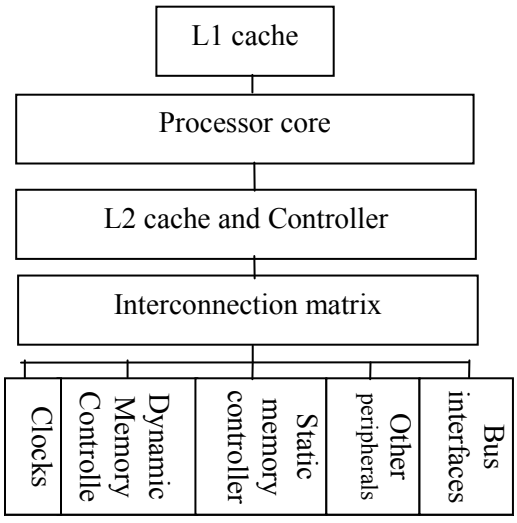

Fig 1. Chip block diagram

To characterize the processor from an energetic point of view, we have used a data logging system implemented on the development board that can measure the voltage and current rapidly on the power lines into the development chip [2]. There are three types of power inputs for the processor, from which we used only two shown in Fig.2.

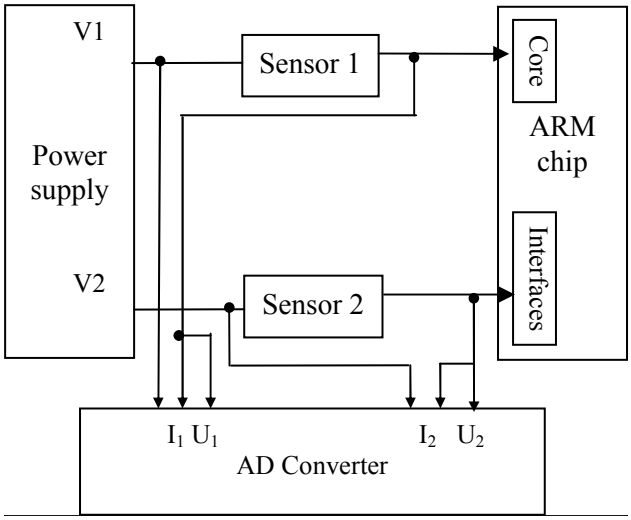

Fig 2. Measuring block schematics

The serially connected resistive sensors convert the current value in differential voltages measured by a AD Converter (ADC). This converter generates binary samples for the voltage and current once each 30 T_{ck} core clock periods.
The measurement results are serially transmitted for storing in memory FPGA hardware registers. The software can interact with these registers and provide values in any computational base (hexa, binary, decimal).
In this case, the sample value is a mean of the current or voltage on the measuring duration T_s.
The core clock for this processor is $T_{ck} \approx 5ns$. The sampling period was chosen to be $T_s = 30\ T_{ck}$ to average out the F_{ck} frequency current spikes existent on the power lines.
In the same time T_s is sufficiently small to make the measurement system sensitive to variations in the current absorbed due to changes in workload on the CPU during application runtime.
Sample values are stored in registers with the help of specialized hardware. Reading these samples is done by the target processor and on the read duration T_w the given application is halted. In order to have the resulting error under a given threshold of ε, (egg.: ε < 1%), we have to limit the number of reads, so as the value T_w to represent less than this threshold (1%) relative to the total process runtime Δt_p:

$$T_w < \varepsilon\, \Delta\, t\, p$$

It is a particular case of the general rule of a measurement system affecting the measured object, and this influence needs to be reduced to an acceptable level [3].
We are also faced with the principial problem of approximating the integral (1) through the rectangular fundamental arias method [4] to obtain the value for the energy absorbed by the processor from the battery (Fig.3).

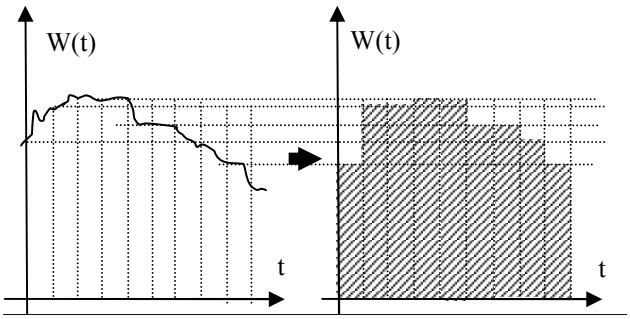

Fig 3. Rectangular fundamentals arias method

The function i(t) not being known, we can't use Nyquist's sampling theorem to determine the time interval between two sampling points in the diagram (an interval which is actually T_w).
A practical solution is to *experimentally* verify how fast the *significant* variations in current are and to choose T_w sufficiently small so as to be able to sense these variations.
We have to mention that this kind of approach is not generally applicable, but it depends on each application in part.

4 Experimental results

We have experimented on the analyzed processor in order to establish how to concretely measure the consumed energy during an application runtime.
The storing capacity of each register is 512 samples on 32 bits.
For the first series of measurements we used a test program (PG1) in which the processor executes simple arithmetic operations inside the core registers, in which we do not access power demanding hardware, and we have compared the individual values of power (current) from three consecutive groups of 512 samples.
In Table 1 we give an example of a group of 8 consecutive samples read from the FPGA registers, for the current/power FPGA consumed by the core for voltage V_1. The peripheral setup and the program content were chosen so the power consumed by the peripherals powered by V_2 to be constant.

Table 1. Sample example

Sample	Stored sample value		
	Current/K_1	Voltage/K_1	Power/K
	Hexa	hexa	hexa
1	01C2	0AD7	00130DEE
2	01C9	0ACB	00134463
3	01D2	0AC2	0013C784
4	01CC	0AE4	001391B0
5	01D3	0ADA	0013CBAE
6	01DA	0AC2	0013EB34
7	01D3	0ADB	0013CD81
8	01CD	0AE1	0013972D

Somehow surprising, the differences between individual measurements are in the order of 1% on the entire duration of the read groups. We have repeated the measurement for a different set of three 512 consecutive sample groups in a different temporal interval and the result was the same.
This first experiment allows us to do a read of the results by the processor after each memory register fill. So we will be able to draw the variation function for current / power through points situated in the interval:

$$T_w = 512 \cdot 30 \cdot T_{ck} \qquad (2)$$

In this case, the error introduced in energetic measurement for the application by the pause necessary for reading the measurement results by the processor was less than 1%.
Using the above observation, the average current / power on T_w can be obtained by considering only a part "n" of the 512 samples, equally distanced in time, instead of the whole 512. In this way we minimize the time required to read the results and the cpu overhead. In Table 2 we present the result of measurements for a group of 512 samples and n=16.

Table 2. Measurement results

Sample	Power/K_1	Sample	Power/K
	Hexa		hexa
1	0014163C	9	00139754
2	0013BD57	10	00146847
3	0013EB34	11	0014778F
4	00140118	12	0013EB34
5	0013B303	13	0013EB88
6	0013EB34	14	0013E88C
7	00146475	15	0013AA24
8	0014805C	16	0013EA68

We assume that on the entire run time for a given application, we register a number of N groups of

512 samples. Then the total energy required to run this program can be calculated with the formula:

$$W = \frac{512}{n} \sum_{j=1}^{N} \sum_{i=1}^{n} 30 \cdot K \cdot p_i \cdot T_{ck} \qquad (3)$$

where: p_i – the current power value for each of the n values of a 512 sample group
k – scaling constant necessary to express the result in I.S. units (Joules)
T_{ck} – core clock period
j – current index of the 512 sample group
i – current index inside a 512 sample group

In the second phase of measurements we meant to compare the energy consumed by the processor core for four different kinds of workload that can appear in typical applications on this processor including power management applications. These are:

a. The processor enters a low power mode (standby) but remains partially active (measuring hardware remains powered on), and interrupts triggers the software to read back the results.
b. The processor is loaded with an application consisting of NOP operations, and reading measurement results is done in a separate thread triggered by interrupts
c. The processor runs the program PG1, described above, and interrupts triggers the software to read back the results.
d. The processor runs the application PG2, which activates a power demanding fast data compression interface, while a second thread reads back results, triggered by interrupts.

Using formula (3) we calculated the energy consumed by the processor on the (fixed) duration of a 512-sample group (2), in conventional unit. The value of the conversion constant K necessary to display results in international system units is less important because the practical purpose for this study is a comparison through a $W(T_1) / W(T_2)$ type ratio for two different implementations of a processor.
For comparison, we retained only the most significant 4 figures from the energy in hexa.
Table 3 synthetically presents the measurement results for the 4 considered cases.
Results are, for the most part, expected based on physics considerations
$W_{b,c,d} / W_a$ in table 3 represents the ratio between the energy measured in cases 'b', 'c', 'd' and the energy measured in case 'a'.

Table 3. Consumed energy comparison

Case	Energy / k		$W_{b,c,d} / W_a$
	Hexa	Dec.	
a	01200	4608	1,000
b	0164F	5711	1,239
c	0165D	5725	1,242
d	01855	6229	1,351

Between case 'a' and 'b' we have observed a difference of about 24%.
The small difference between cases 'b' and 'c' is explained by the fact that in both the case of the PG1 program and the NOP loop the sub-blocks activated within the core a similar in terms of power drain.
In case d. the energy absorbed by the core is 35% greater than standby due to the activation of a high frequency large bus interface.

We expected a bigger difference towards the standby mode, but for the given processor, standby leaves many hardware sub-blocks still active (as opposite to more advanced low power modes available). The active hardware runs at core frequency, so the fast transitions contribute to the relatively high power demand in standby mode.
Also, the share of leakage currents in the total energy consumed by all the logic is more important in circuits powered from lower(1.2V) voltage supplies, than older 2,5 or 3,3V, and this represents a constant for all the studied cases.

5. Conclusions

1. The method presented in this paper, supported by experimental results, permits the objective comparison on *energetic* efficiency for different hardware loads (variants) of a processor powered by battery, and for the case when the application run time are different in these cases. Comparison through *power* is inadequate in this case.

2. The presented measurement results give useful in formations in case of using power management on a modern ARM architecture.

3. In this paper it has been succeeded to reduce the influence of the measuring system over the measured object and implicitly the error underneath the desired threshold of 1%. Having in mind that the target processor also handles the measuring system, this is a good performance. And for practical applications this level of influence is completely acceptable.

ACKNOWLEDGMENT

The work of this paper was done with support from OPTIMTRAF 71-114/2007 project, INTELPROT 11-069/2007 project and SINERG/2008 project, .

References:

[1] ARM-The Arhitecture for the Digital World, www.arm.com
[2] User Guide HBI-0147, Paper of ARM Limited, 2007, ARM DUI0425B
[3] Costin Stefanescu, Nicolae Cupcea, Electronica Aplicata-Sisteme Inteligente hardware-software de masurare si control, Editura Albastra, Cluj-Napoca, Romania, 2002
[4] Ioan Rusu, Metode numerice cu aplicatii in limbaj C, Matrixrom, Bucuresti, Romania, 2008

But when we factor in power efficiency, things get crazy. I gave my Ryzen and Apple processors the benefit of every possible doubt when generating the above chartsâ€”I used core power (not total package power) on the Ryzen 4700U and ran tests with the Gnome3 desktop shut down. For the Apple, I only had access to whole-system power draw, so I subtracted the "desktop idle" power draw from the "under test" power draw.Â Micro Magic intends to offer its new RISC-V design to customers using an IP licensing model. The simplicity of the designâ€”RISC-V requires roughly one-tenth the opcodes that modern ARM architecture doesâ€”further simplifies manufacturing concerns, since RISC-V CPU designs can be built in shuttle runs, sharing space on a wafer with other designs. I usually read that RISC processors usually have a lower power consumption than CISC processors for example: ARM implementations (1-2 W) to x86 implementations (5 - 36 W). Does that apply for all the different RISC and CISC processors, or there are some excepltions. Why this big difference? architecture processor microprocessors.Â RISC assumes that computer has finite number of transistors and what are the most number of instructions that can fit onto the computer. That is not the case for CISC. Look at Coursera course on computer architecture particularly Lecture 1 part 5(L1s5). This approach modifies a RISC processor by integrating an additional Fetch Look-Aside Buffer (FLAB) for instructions. While the first fetch of any instruction results in normal execution, this instruction is combined in parallel with former instructions for later execution and saved inside the FLAB. The architecture works like a dynamic Very- Long-Instruction-Word architecture using code morphing. Extensive simulations indicate that this approach results in average instructions per cycle rate up to 1.4. The more important fact is that these values are obtained at moderate hardware extensions. The Space-Time-Efficiency E is defined and shows values from 0.5 to 1 for all modified architectures, relative to the RISC processor. Keywords. RISC processor design has separate digital circuitry in the control unit, which produces all the necessary signals needed for the execution of each instruction in the instruction set of the processor. Examples of RISC processors: IBM RS6000, MC88100. DECâ€™s Alpha 21064, 21164 and 21264 processors. Features of RISC ProcessorsÂ Most of the RISC processors are based on the hardwired control unit design approach. In hardwired control unit, the control units use fixed logic circuits to interpret instructions and generate control signals from them. It is significantly faster than its counterpart but are rather inflexible.Â Most instructions in a RISC instruction set are very simple that get executed in one clock cycle. 5. RISC processors use simple addressing modes.