Energy Estimation and Optimisation of Embedded Systems using Cycle Accurate Simulation

Ana Abril,
Habib Mehrez
and Frédéric Pétrot
ASIM/LIP6 Lab, University Paris VI, France
Email: Ana.Abril@lip6.fr

Jean Gobert
and Carolina Miro
Philips Digital Systems Laboratory,
51, rue Carnot - B.P. 301
Suresnes, France

***Abstract*— This paper proposes a method for energy consumption estimation and optimisation on hardware-software embedded systems-on-chip. The starting point is an architectural description of the system used for simulation, that employs very abstract C-based models of the hardware components. We focus on the cycle-accurate level to improve the estimation accuracy. Behavioural models are extended with energy models that take into account the operations executed per transition into the state machine of the components. The method has been tested in a MPEG4 decoder implementation. The error of the energy estimations has been estimated lower than 6% from physical measurements. Low power techniques have been applied and analyzed like another memory hierarchy, clock gating, voltage/frequency scaling, and some others. They allow to reduce the consumption cost of the system in 93%.**

I. Introduction

The reduction of power consumption in integrated circuits demands the application of low power techniques in all levels of the design flow, but the most important savings are gained at the highest levels. This paper focuses on the architectural cycle-accuracy level. This allows a good accuracy for a very fast simulation time.

A new methodology for analysis and reduction of the power consumption of SoC, at the earlier stages of the system architecture definition is proposed. A simulation of the complete system is performed with a cycle-accurate system simulator. It simulates the system using functional models of the hardware blocks, the behavior of each block being described as a state machines with data-path. The energy estimation is performed by adding an energy view into the models for every hardware block. It consists of an estimation of the energy consumption of each operation executed per transition on the state machine. Then the system is simulated with a set of representative input data to obtain the power estimation. We propose an approach for creating the energy models of all blocks of the system, including new hardware blocks for which we do not have a low level implementation.

A brief discussion of the existing power estimation approaches is given in the section 2. Section 3 explains our approach and introduces the simulation tool we used. In section 4, we describe the system model detailing how to model the energy for each category of component. Section 5 illustrates the MPEG4 system exemple, the results for energy estimation and the optimizations applied. A conclusion is given in last section.

II. Related Research

Higher level energy estimations of embedded systems are proposed by some approaches [2] [3] [4] [5] [6]. Most of them choose the simulation of the software component of the system using an ISS (Instruction Set Simulator), enriched with power models of the processor's instructions [2] [3] [4]. Some others use high level RTL simulators also incorporating power models for the estimation of the hardware dissipation [4] [5]. Hardware-software co-estimation is performed linking these two kinds of simulators [3] [4]. If the power models are well defined, these methods give a good accuracy, but they are quite slow because they use the RTL simulation. At the algoritm level, Orinoco [6] allows to compare the power consumption of different algorithms running in different architectures.

Each one of these methods covers one stage on the highest levels of the design flow. However none enables to estimate and optimize the energy dissipation of all elements of an architectural description of an embedded system. A complete approach allowing this should *a)* use concurrent and very fast simulation, *b)* provide the energy dissipation of all the components including new accelerators and the interconnections with enough accuracy and *c)* give the energy dissipation evolution of the system, to experiment dynamic low power techniques at system level.

In this paper, we present a new approach that, using some ideas from previous approaches, attempts to gather all these needs.

III. Proposed Approach

An embedded system is a complex combination of components like processors, coprocessors, memories and interconnections created to execute a given application (see Fig. 1).

To develop the system architecture and the communications between blocks, and to support the software development of low level drivers, a system simulation using functional models of the blocks is required at the beginning of the design flow. These models need to be sufficiently abstract because billions of simulation cycles are required. The architectural simulation performed at cycle level allows a very fast analysis because

the simulation speed and the performance accuracy are very high for a short modelling time.

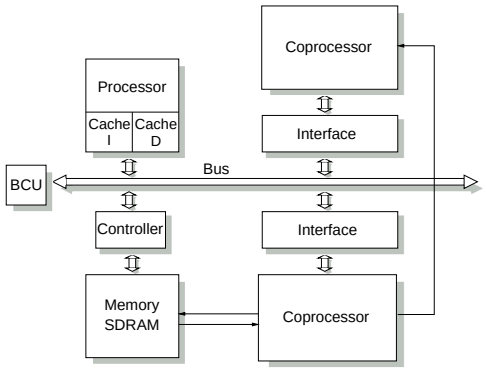

Fig. 1. Architecture of an embedded system.

There exist several modelling methods at architectural level like TSS [1], CASS [7], and recently SystemC. They are all event-driven. If the event is the clock, they perform cycle-accurate simulation. Components are modelled as states machines where the clock produces the state transistion. The simulation method is different for each one of these tools, but the modelling principle at cycle level is the same.

We use TSS, a cycle-accurate C-based simulation framework developed by Philips [1]. TSS also enables simulations with other tools, allowing to integrate Instruction Set Simulators (ISS), VHDL, Verilog and TSS models. The TSS simulator performs cycle-accurate functional simulation. In this paper we present how the TSS models can be enriched with an energy view allowing to get energy estimations using cycle accurate simulations. The solution proposed consists on analyzing the basic functional states of the state machine of each TSS model, in order to estimate energy consumption per transition. These energies per transition are accumulated during the simulation of a particular application, giving the energy estimation per component and on the entire system at point in time. This could be not a completely accurate estimation (it depends on the models accuracy) but it could be sufficient to find the best hardware architecture scenario in terms of power consumption. This method can be generalized for all others simulators that model the components at cycle level like states machines like SystemC.

IV. System and Components Models

The energy consumption for executing a task in the system can be observed at several levels. We analyze these from higher to lower level. At the first level we can consider the total energy dissipation for the execution of the whole application task in the target architecture. If m is the total number of components of the system, the total energy consumption is obtained by accumulating the energies of all the components, according to (1).

$$E_{total} = \sum_{i=1}^{m} E_{component_i} \quad (1)$$

At a second level we consider the total energy consumption per component, that is the accumulation of the energy dissipation for all the transitions in the state machine of the component during the execution of the application. If it needs n transitions, this energy is represented by the equation 2 (the number of transitions and cycles is the same according to the state machines definition).

$$E_{component} = \sum_{j=1}^{n} E_{component,transition_j} \quad (2)$$

Each component can have one or more state machines to simulate its behaviour. To simplify the notation, we have considered only one state machine per component, but the method remains valid in all cases. The transitions between states represent the functionality of the component per cycle. During each transition, one or more basic operations are executed: read, write, operation1, operation2, wait, etc. For r operations, this energy is represented by the equation 3

$$E_{transition} = \sum_{k=1}^{r} E_{operation_k} \quad (3)$$

The energy associated to each operation of a component $E_{comp,op}$, represents the lowest level of our model, and it corresponds to the dynamic and static energy dissipation during the execution of the operation. The set of energies per operation for x possible operations is called the energy model of the hardware block and is given in the equation 4.

$$E_{component} = \{E_{comp,op1}, E_{comp,op2}, ..., E_{comp,opx}\} \quad (4)$$

In this way we define the energy dissipation per transition as a function of the operations executed during the current transition in the state machine. An example for n transitions is showed in the equations 5.

$$\begin{cases} E_{component,transition_1} = E_{comp,op_1} \\ E_{component,transition_2} = E_{comp,op_4} + E_{comp,op_6} \\ ..... \\ E_{component,transition_n} = E_{comp,op_x} \end{cases} \quad (5)$$

During simulation, each component switchs per cycle in a new or the same state, depending on its functionality and inputs. In fig. 2, we see the state machine of a component example, with the energies per operation associated to each transition.

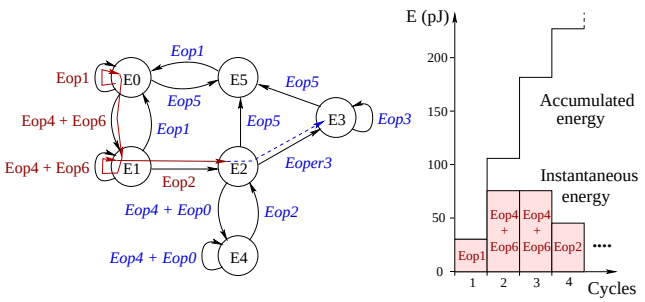

Fig. 2. State machine of a component example

The energy values per operation are calculated using several methods: *1) macro-modelling*, giving a library of values measured at low level, *2) fast-modelling*, using equations where parameters are easily estimated from technological, structural and functional information and, *3) entropy*, estimating the completely unknow parameters using entropy equations.

The processor and memory energy values are calculated using macro-modelling and the values come from datasheet information. Hardware accelerators and interconnect use fast modelling and the parameters of equations are estimated from low level information or using entropy. Next subsections will explain in detail the energy models and the values per component.

A. Processor energy model

The processor can execute two kinds of operations per cycle: the *actives* ones during the execution of instructions, or the *idle* ones when there is a cache miss and processor stalls; and both in different modes of working: *nominal*, *voltage/frequency scaling* and *sleep*. Nominal mode is the normal one, the scaling mode reduces the frequency and voltage nominal values, and sleep mode employes clock gating techniques. The energy values per cycle can be deduced from datasheets or can be measured at physical level per instruction and mode of working.

B. Memory energy model

The memory energy model corresponds to the energy consumption per cycle for each basic functionality of the device. SDRAM memories are complex and big structures where at each transition, an internal command is generated to control the structure (e.g. idle, activate, read,...). We associate an energy value to the transition of each command in the corresponding TSS model. For the others memories (SRAM, FIFO, etc), the energy model is the same but only using *read*, *write* and *idle* commands.

The energy values per command are calculated from the voltage, current and frequency nominal values given in datasheet, and scaled to current conditions of voltage using the Eq. 6 and 7.

$$E_{nom} = \frac{I_{nom} \times V_{nom}}{f_{nom}} \tag{6}$$

$$E = E_{nom} \times \left(\frac{V}{V_{nom}}\right)^2 \tag{7}$$

C. Hardware accelerators energy model

Hardware accelerators are modelled like state machines where different operations are executed for each state transition. Each operation has associated an energy dissipation. Their values are calculated using fast modelling equations. The total energy dissipation per operation is the addition of the combinatory energy (gates) and the sequential energy (flip-flops). The combinatory energy E_{comb} is calculated using the Eq. 8.

$$E_{comb} = \mu_g \times E_{gate} \times N_{gates} \tag{8}$$

where μ_g is the gates activity (% of switching gates), E_{gate} the energy consumption per switching gate for a technology and N_{gates} the total number of equivalent gates in the component. The sequential energy E_{seq} is calculated using the Eq. 9.

$$E_{seq} = [\mu_{ff} \times E_{ff} + (1 - \mu_{ff}) \times E_{ffck}] \times N_{ff} \tag{9}$$

where μ_{ff} is the flip-flops activity (% of flip-flops with an output transition), E_{ff} the flip-flop active energy dissipation (the total energy required by a flip-flop to perform an output transition in a clock edge), E_{ffck} the flip-flop clock energy dissipation (the energy dissipated by a flipflop in a clock edge without an output transition), and N_{ff} the total number of equivalent flip-flops. The parameters of the equations are obtained from low level measures or using entropy. We estimate these values for all the working modes of the component: *nominal*, *voltage/frequency scaling* and *sleep* (clock gating).

D. Interconnect energy model

An embedded system can perform different types of transfers. The energy model is the same for all of them and is made using fast-modelling. It consists of the energy dissipated for all the switching lines per cycle. If we have N_{switch} switching lines during the transition, with a capacitance C_{line} per interconnect line and a voltage switch of V_{dd}, the energy used in the model is given by Eq. 10.

$$E_{interconnect} = N_{switch} * C_{line} * V_{dd}^2 \tag{10}$$

V. MPEG4 IMPLEMENTATION

This approach has been tested on an example of embedded system for portable application: a MPEG-4 video decoder, implementing the simple profile. A preliminary HW/SW partitioning has been already made. The system consist of an ARM processor with I+D caches, an SDRAM memory, hardware accelerators and interconnect.

The energy model of the processor is based on Sinha work [8], that demonstrated that the energy consumption in ARM processors varies only about 8% between the various instructions of a program. Therefore we can consider an average energy per cycle for all the instructions of our application program. The TSS state machine detects the active and idle cycles per transition, and accumulates the corresponding energy depicted in ARM data-sheet.

The memory used is a Micron 64x32 Mbits DDRC SDRAM memory, at 2.5 V and 83 MHz. The energy values per command are deduced from datasheets information and inserted in the TSS model.

Several accelerators are used too, each one performing several operations per cycle. The energies per operation are calculated using parameters calibrated from RTL measurements on a CMOS12 implementation and from entropy estimations.

Two kinds of interconnect are used: onchip and offchip. The capacity and voltage parameters of each one are deduced from other implementations, and the number of switching lines are detected using spy models in the TSS simulation. An example of the energy values is showed in the table I.

TABLE I

THE ENERGIES PER OPERATION ON THE SYSTEM

Block	Mode	Energy (pJ)
Processor (ARM940T + caches)	Active nominal (1.3V)	250
	Idle nominal (1.3V)	110
	Active V/F different (1V)	150
	Idle V/F different (1V)	60
	Sleep (1.3-1V)	10
Block	**Command**	**Energy (pJ)**
DDRC SDRAM (Micron)	Precharge standby	1407
	Precharge power down	87
	Operating	2993
	Active standby	1485
	Active power down	860
	Read	4610
	Write	3438
	Precharge	1497
	Refresh	4594
Block	**Parameters**	**Value**
IDCT (Inverse Cosinus Discrete Transform)	N_gates	6180
	N_ff	440
	E_gate (CMOS12)	10 fJ
	E_ff (CMOS12)	53 fJ
	E_ffck (CMOS12)	19 fJ
	μ_idct1D	36 %
	E_oper_idle	8.269 pJ
	E_oper_idct1D	41.253 pJ
	E_oper_sleep	1.26 pJ
Block	**Parameters**	**Value**
Interconnections	wire_capacity	1.1 pF
	voltage	1.2 V
	E_line_AHB_bus	1.6 pJ

VI. EXPERIMENTAL RESULTS

We have decoded an INTRA images sequence in our TSS MPEG4 decoder implementation example. The format is QCIF (176 pixels/line $\times$ 144 lines at 15 frame/s). The energy estimation results for one image decoded in the normal working mode are showed in the table II. The accuracy of these estimations depends on the accuracy of the cycle-accurate functional modelisation and on the accuracy of the energy values. Our TSS models are well defined and the energy values are calibrated from RTL measurements, so the error of the energy estimation is considered lower than **6%** from physical measurements.

The energy results show that the SDRAM memory is the most power consuming element, after the processor, then the off-chip interconnect and finally the accelerators. This is because it is a very big and consuming memory, and their off-chip accesses are quite slow, giving a lot of idle cycles in all elements. To reduce this energy we apply low power techniques at architectural level, specially on the memory. The more interesting techniques are another memory hierarchy, data embedded compression before memory stores, clock gating for all elements, several frequencies between the components and voltage reduction on the processor. The results obtained using these techniques on each one of the components are showed in the 3rd column of the table II. We observe very important energy reductions in almost all the components. Techniques are applied by components but the effect influences all the elements on the system and our approach allows to study this effect. The maximum reduction obtained when all these techniques are applied together is **93%**. This is a very good result that confirms the interest of our approach for energy estimation and optimization at architectural level.

TABLE II

THE ENERGY CONSUMPTION OF A MPEG4 DECODER

Component	Image Energy (normal mode)	Low Power Technique	Percentage Reduction
ARM	641 μJ	Voltage reduction	60 %
Sdram	5725 μJ	Embedded SRAM	91 %
HW decoder	46 μJ	F/Vdd reduction	90 %
Off-chip interconnect	366 μJ	Embedded SRAM	100 %
On-chip interconnect	6 μJ	-	0 %
Total system energy	**6784 μJ**	**475 μJ**	93 %
Nb total cycles	3236532	2469759	
Frequency	83 MHz	83 MHz	
Total system power	**174 mW**	**16 mW**	91 %

VII. CONCLUSIONS

We have presented an approach to estimate and optimize the energy consumption in architectural level descriptions of embedded systems. We use the cycle-accurate simulation where functional models of the hardware components are extended with an energy view. This consists of the energy per operation for each state transition of the component. The approach has been validated on a hardware/software MPEG-4 video decoder system example, obtaining energy estimations during simulation. The results gives a very good accuracy (6%) because the functional models and the energy numbers are sufficiently accurate. We have applied several low power techniques and observe very important energy reductions (93%). It demonstrates the application of our approach for the fast and early energy evaluation and optimizations of systems-on-chip at architectural level. This method can be extended to others system simulation environments at cycle level like SystemC.

REFERENCES

[1] F. Theeuwen, "TSS, System Simulation at Philips Research". *Talk at the MEDEA Workshop on System Simulation*. May 1998.

[2] T. Simunic, L. Benini and G. De Micheli, "Cycle-Accurate Simulation of Energy Consumption in Embedded Systems". *Proceedings of the IEEE 36th DAC, Design Automation Conference*, 1999, pages 867-872.

[3] J. Henkel and Y. Li, "Avalanche: an environment for design space exploration and optimization of low-power embedded systems". *Transactions on VLSI Systems*, volume 10, number 4, 2002, pages 454-468.

[4] M. Lajolo, A. Raghunathan, S. Dey and L. Lavagno, "Efficient Power Co-Estimation Techniques for System-on-Chip Design". *Proceedings of the IEEE DATE, Design Automation and Test in Europe conference*, 2000.

[5] BullDast, "PowerChecker: An integrated environment for RTL power estimation and optimization". Version 4.0, http://www.bulldast.com.

[6] ChipVision, "Orinoco: A high-level power estimation and optimization tool suite3. http://www.chipvision.com.

[7] D. Hommais, F. Petrot and I. Auge, "A Toolbox to Map System Level Communications on HW/SW Architectures". *Proceedings of the 12th International Workshop on Rapid System Prototyping*, 2001.

[8] A. Sinha and A.P. Chandrakasan, "JouleTrack - A web based tool for software energy profiling". *Proceedings of the IEEE 38th DAC, Design Automation Conference*, 2001, pages 220-225.