Energy estimation and optimization in architectural descriptions of complex embedded systems

Ana Abril[a], Habib Mehrez[a], Frédéric Pétrot[a], Jean Gobert[b] and Carolina Miro[b]

[a] ASIM/LIP6 Lab, University of Paris VI, 5, Place Jussieu, 75005 Paris, France ;
[b] Philips Digital Systems Laboratory, 51, rue Carnot BP 301 Suresnes, France

ABSTRACT

This paper proposes a method for energy consumption estimation and optimisation on hardware-software embedded systems-on-chip. The aim of our work is to provide a simulation framework enabling power estimations of high level descriptions (behavioural C models) of systems that include all the hardware components also the new ones. Such analysis are needed to select the best hardware architecture and software organization for a particular application in terms of power consumption and to apply low power techniques at system level.

The starting point is the architectural description of the system used for simulation. It 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 machines of the components.

The method has been tested in a MPEG4 decoder implementation. The error of the energy estimations was estimated lower than 6% from physical measurements. Low power techniques were applied and analyzed like another memory hierarchy, clock gating, voltage/frequency scaling, and some others. It has permitted to reduce the consumption cost of the system on 93%.

Keywords: energy estimation, low power, hardware-software co-simulation, embedded system, System on Chip, cycle-accurate, energy model

1. INTRODUCTION

The reduction of power consumption on 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. At design start time, important choices are made that will significantly influence the power consumption of the system because they define the type and utilization of hardware components: Use of general purpose processors and/or dedicated coprocessors, memory hierarchy, type of interconnects, etc.

This paper focuses on the architectural cycle-accuracy level. A new methodology for analysis and reduction of the power consumption on SoC at architectural level is proposed. This level gives a good accuracy for a very fast simulation time. The simulation of the complete system is performed using a cycle-accurate system simulator that employs functional models of the hardware components to describe the system. The behavior of each component is described in C-language as a state machine with data-path. Our experimentation is performed using TSS, an event-driven simulator developed by Philips.[1] Nevertheless, the proposed method can be use with another cycle-accurate simulation environment like SystemC.

The energy estimation is performed by adding an energy view into the functional model of every hardware component. This energy view consists of an estimation of the energy consumption dissipated by each operation executed per transition into the state machine of the component. These energy values are included into the simulation model without slowing down the simulation speed. That way, when the system is simulated with a set of representative input data, energy estimations per component and for the whole system are obtained. We

Further author information: (Send correspondence to Ana Abril)
Ana Abril: E-mail: Ana.Abril@lip6.fr, Telephone: +33 (0)1 44 27 71 73
Habib Mehrez: E-mail: Habib.Mehrez@lip6.fr, Telephone: +33 (0)1 44 27 47 61

propose an approach to create the energy models of all the components of a typical embedded system, including the new hardware blocks for which we do not have yet 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, and section 6 gives the results for energy estimation and optimization. A conclusion is given in the concluding section.

2. RELATED RESEARCH

A common approach to obtain fairly accurate power estimations of large circuits is to use a conventional RTL simulator that models power consumption of the basic cells with parameters extracted from circuit level simulations (SPICE, PowerMill) or from the characteristics of a standard cell library. This information is then embedded in the VHDL description of every basic cell so that the logic simulation allows to obtain the power consumption of the circuit. DIESEL,[2] the power estimation tool developed in Philips, uses this approach. This method has some drawbacks. It takes place in an advanced phase in the design process so it can be used only for analysis of an implementation and not for making choices that will optimize power consumption. Moreover the simulation time for large circuits becomes very important making rather impracticable power estimations of a complete hardware/software system.

Higher level energy estimations of embedded systems are proposed by some approaches, like the Simunic's one,[3] Avalanche,[4] Lajolo,[5] Powerchecker[6] or Orinoco.[7] 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 (Simunic's,[3] Avalanche[4] and Lajolo[5]). Some others also use high level RTL simulators incorporating power models for the estimation of the hardware dissipation (Powerchecker[6] and Lajolo[5]). Hardware-software co-estimation is performed linking these two kinds of simulators (Avalanche[4] and Lajolo[5]). If the power models are well defined, these methods give a good accuracy, but they are quite slow because they use the RTL simulation. The use of co-simulation improves the accuracy because it allows to obtain the interactions between modules during their execution. Some of these methods give not only the total energy consumption at the end of the simulation, but also its evolution in time (Lajolo[5]). Orinoco[7] is the highest level estimation tool we know (algorithm level), because it 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 be sufficiently:

1. `Powerful`, to allow power modeling of all the components of the SoC, and gives the total energy and the energy evolution in time during simulation,
2. `Fast`, to allow energy analysis of a whole SoC in a reasonable time,
3. `Accurate`, to allow the energy comparison of several architectures and the application of dynamic low power techniques,
4. `Flexible`, to allow dynamical manipulation of power parametres,
5. `Easy to use`, to not increase development time of designers.

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

3. PROPOSED APPROACH

An embedded system is a complex combination of components like processors, coprocessors (or hardware accelerators), memories and interconnections, connected as in the exemple on Fig. 1, and created to execute a given application. Its design starts writing the algorithm that performs the application, following by the partitioning of the algorithm functions in hardware and software tasks, and the choice of the hardware components to execute the application in a particular architecture.

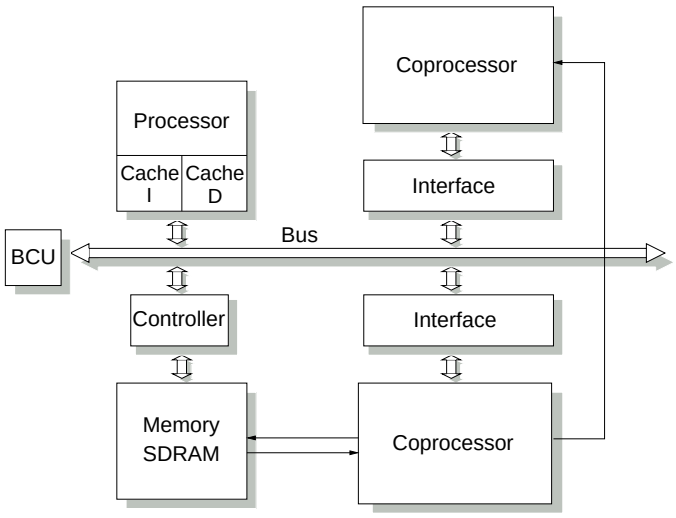

Figure 1. Architecture of an embedded system.

At this time, a system simulation is required to develop the system architecture and the communications between blocks and to support the software development of the low level drivers. That can be performed using functional models of the components. Those 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. Consequently, this simulation allows the exploration of the whole system giving the possibility to find and solve problems on the architecture design directly from the source. An exemple is showed in the paper of Jang.[8]

Our approach for the energy estimation consists of adding to the cycle level simulation a power consumption estimation capability. It will allow to obtain power numbers of a complex embedded system very soon on the conception flow and giving a good arrangement between accuracy, modelling time and simulation speed. There exist several modelling methods at architectural level like TSS,[1] CASS,[9] and recently SystemC.[10] They are all event-driven. If the event is the clock, they perform cycle-accurate simulation. In these approaches, the hardware components are modelled as states machines where the event (clock edge) produces the state transition. Even if the simulation method and models description of the components are different for each one of these tools, 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 is used for 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 is to analyze the basic functional states of the state machine of each TSS model in order to estimate energy consumption per transition. An energy value is associated to each state transition of the FSM (Functional State Machine) of the component. This energy value represents the dissipation of the operations executed in the corresponding cycle. These energies per transition are accumulated during the simulation of a particular application, giving the energy estimation per component and for the entire system

at point in time. This is not a completely accurate estimation but it is 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. The whole energies per transition of a component constitute its energy model. The way to build the energy models for all the components and to add it to the TSS models is described in the next section.

4. 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).

$$E_{total} = \sum_{i=1}^{m} E_{component_i} \tag{1}$$

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 thanks to a property of the state machines definition that says that at every cycle there is always a transition for a new or the same state).

$$E_{component} = \sum_{j=1}^{n} E_{component,transition_j} \tag{2}$$

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. For each transition, one or more basic operations are executed: read, write, operation1, operation2, wait, etc. Each operation has dynamic and static energy consumptions associated. The energy associated to each operation of a component $E_{comp,op}$, corresponds to the lowest level of our model. 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 3.

$$E_{component} = \{E_{comp,op1}, E_{comp,op2}, ..., E_{comp,opx}\} \tag{3}$$

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 4.

$$\begin{cases} E_{component,transition_1} = E_{comp,op_1} \\ E_{component,transition_2} = E_{comp,op_4} + E_{comp,op_6} \\ ..... \\ E_{component,transition_n} = E_{comp,op_x} \end{cases} \tag{4}$$

In an example of embedded system like the showed in Fig. 1, the E_{total} can be written as in (5).

$$E_{total} = E_{processor} + E_{sdram} + E_{coproc1} + E_{coproc2} + E_{bus} \tag{5}$$

During simulation, each component is switching 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.

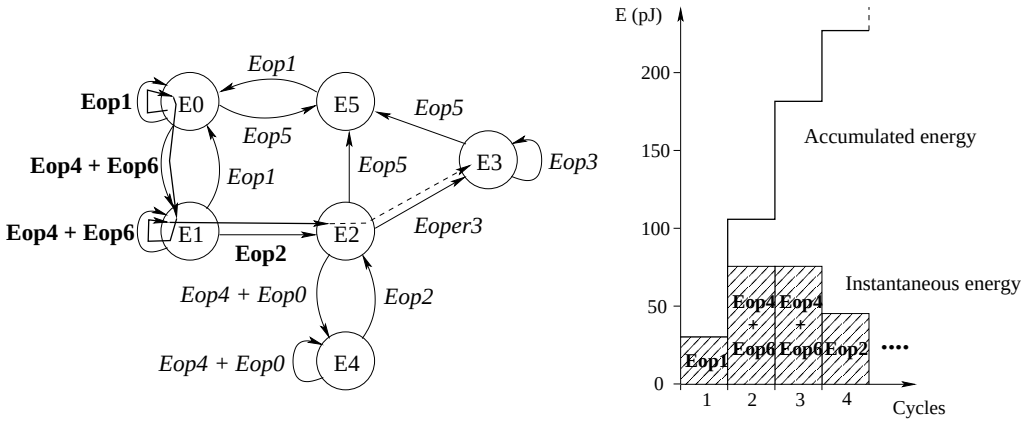

Figure 2. State machine of a x component example

In fig. 2, the accumulation of the energies corresponding to the operations executed in the current state transition, is calculated giving the instantaneous energy per cycle of the component. The detail of this calculation is showed in the following equations 6.

$$\begin{cases} E_{comp_x,transition_1} = E_{comp_x,oper_1} \\ E_{comp_x,transition_2} = E_{comp_x,oper_4} + E_{comp,oper_6} \\ E_{comp_x,transition_3} = E_{comp_x,oper_4} + E_{comp,oper_6} \\ E_{comp_x,transition_4} = E_{comp_x,oper_2} \\ ..... \end{cases} \tag{6}$$

The accumulated energy gives the total energy per component with the equation 7.

$$E_{comp_x} = \overbrace{E_{comp_x,oper_1}}^{E_{comp_x,transition_1}} + \overbrace{E_{comp_x,oper_4} + E_{comp_x,oper_6}}^{E_{comp_x,transition_2}} + \overbrace{E_{comp_x,oper_4} + E_{comp_x,oper_6}}^{E_{comp_x,transition_3}} + ... \tag{7}$$

If there are m components in the system, the total energy dissipation is the addition of the accumulated energies of each one, like shows the equation 8.

$$E_{totale} = \overbrace{E_{comp_x} + E_{comp_y} + E_{comp_z} + ... + E_{comp_m}}^{m \text{ components}} \tag{8}$$

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.
3. **Entropy**, estimating the completely unknow parameters of the new hardware coprocessors 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*.

The *energy estimation flow* proposed is showed in fig. 3. It begins with the parallel description of the partitioned architecture at system level (C/C++, YAPI, SystemC). The task are associated to processors and

hardware accelerators, and the hardware components are described at cycle level in C-models (TSS, SystemC, CASS). Some C-models can be already available like the memories or general processors.

The energy dissipation per state transition of each component is calculated and added to the model in a well defined way according to the simulator functionning. The simulation starts with the corresponding current energy parameters and inputs, and simulates the application behaviour. During the simulation and at the end, the energy accumulated values can be observed and analysed. If there are all satisfactory, the conception flow continues and the hardware description begins. Otherwise, several solutions can be proposed to correct the power consumption results:

- Keeping the same partitionning but changing the model and system parameters (i.e. memory size, data/bus width, frequency, voltage, etc).
- Keeping the same partitionning but modifying the functionnals models (i.e. including low power states).
- Changing the interfaces, the partitionning and the choices of architecture.

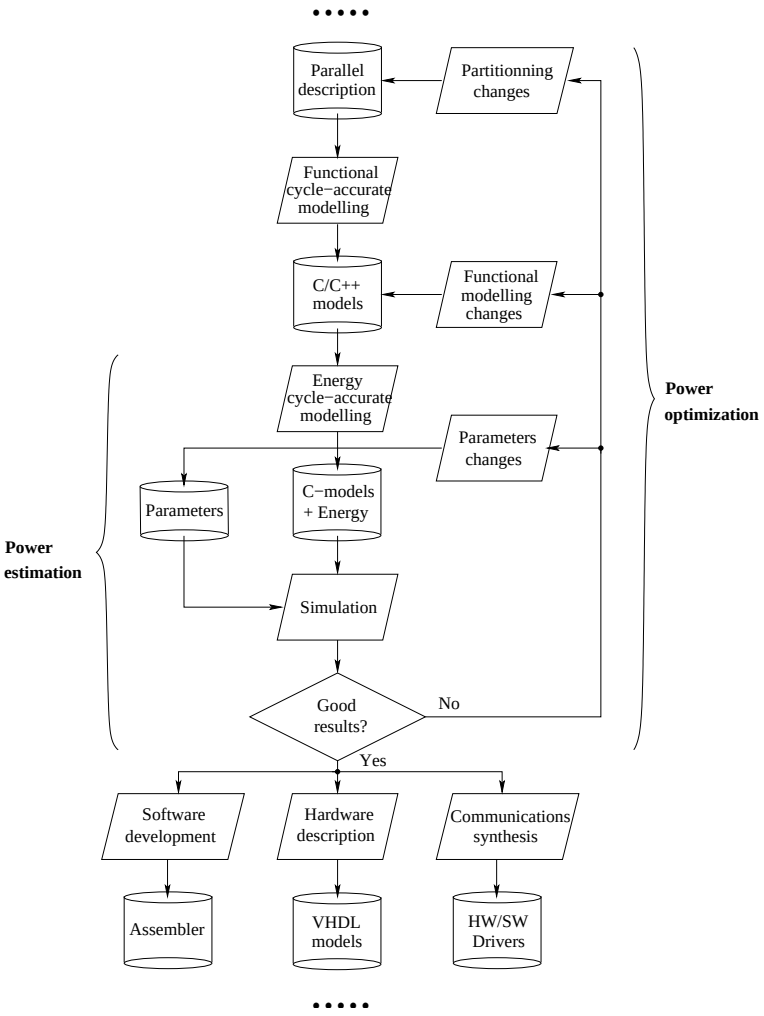

Figure 3. Energy estimation flow at cycle level

Next subsections will explain in detail the energy models giving some energy values per component as exemple.

4.1. 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.

4.2. Memory energy model

The memory energy model corresponds to the energy consumption per cycle for each basic functionality of the device. SDRAM memories are very 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. 9 and 10.

$$E_{nom} = \frac{I_{nom} \times V_{nom}}{f_{nom}} \tag{9}$$

$$E = E_{nom} \times \left(\frac{V}{V_{nom}} \right)^2 \tag{10}$$

where I_{nom}, V_{nom} and f_{nom} are the nominal values given in datasheets, and V the current voltage value used in the system.

4.3. 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. All these energies constitute the energy model of the component. Their values are calculated using fast modelling equations from the general equations of dynamic power consumption on digital CMOS circuits. These equations use structural, functional and technological parameters, that can be obtained from low level measures, or can be estimated using the available information about the component or using techniques like the entropy.[11]

The energy value is the addition of the dynamic and static energies. The dynamic energy corresponds to the switching activity in transistors and The total energy dissipation per operation is the addition of the combinatory energy (gates) and the sequential energy (flip-flops). The static energy corresponds to the leakage currents in transistors. Then the total energy consumption per operation is expressed as in the Eq. 11.

$$E_{total} = E_{comb} + E_{seq} \tag{11}$$

The combinatory energy E_{comb} is calculated using the Eq. 12.

$$E_{comb} = \mu_g \times E_{gate} \times N_{gates} \tag{12}$$

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. 13.

$$E_{seq} = [\mu_{ff} \times E_{ff} + (1 - \mu_{ff}) \times E_{ffck}] \times N_{ff} \tag{13}$$

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 static energy is negligible during the active cycles. However

it has to be account in idle cycles, especially in submicron technologies, where the static currents, I_{fuite}, become very important. Its value is calculated using the equation 14.

$$E_{stat} = \frac{1}{2} \times (4N_{gates} \times 16N_{ff}) \times I_{leakage} \times V_{dd} \times \Delta T \tag{14}$$

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).

4.4. 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. 15.

$$E_{interconnect} = N_{switch} * C_{line} * V_{dd}^2 \tag{15}$$

5. MPEG4 IMPLEMENTATION EXAMPLE

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. It is illustrated in the fig. 4

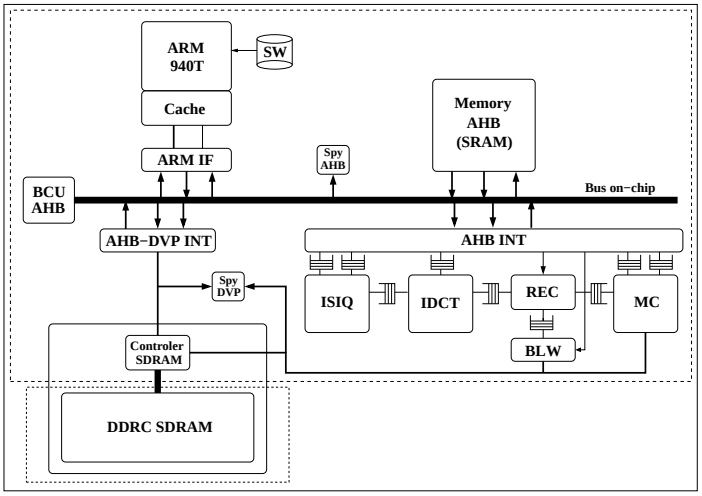

Figure 4. MPEG4 system example

The energy model of the processor is based on Sinha work,[12] 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 1.

An example on our system is the IDCT, an accelerator of the MPEG4 decoder. It can perform two types of operation per transition: *idle* and *idct 1 Dimension* (one or two times per transition). The values of their parameters and the energies per operation in Philips CMOS12 implementation can be found in the table 1. There are others accelerators in our system and their energy models are very similar to this one, but adapted for their particular operations.

A TSS spy model detects the number of switching lines per cycle in the interconnections on the system. An example of values for the amba-AHB bus is showed in the table 1.

Table 1. 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

6. 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 2. 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 percentage of power reduction obtained using these techniques are showed in the 4rd column of the table 2 for each components and in the fig. 5 for the whole system.

Table 2. 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 energy	**6784 μJ**	**475 μJ**	93 %
Nb total cycles	3236532	2469759	
Frequency	83 MHz	83 MHz	
Total Power	**174 mW**	**16 mW**	91 %

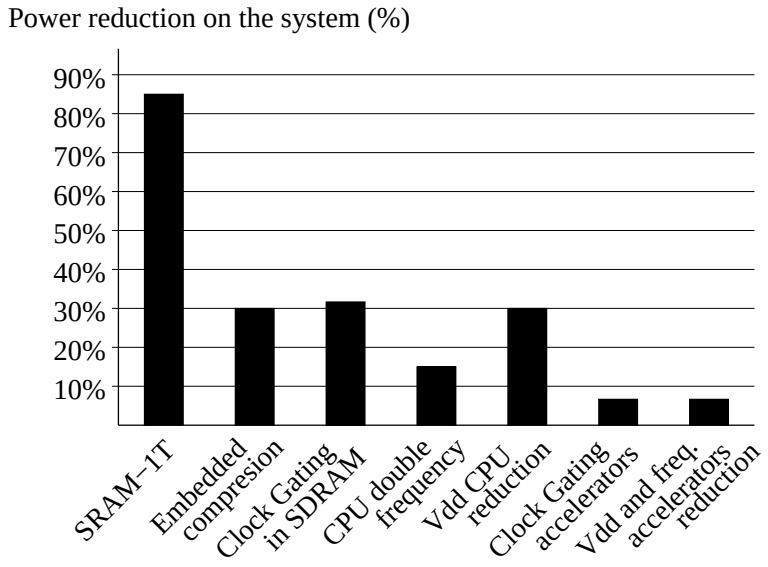

Figure 5. Percentage of power reduction on the system per technique

We observe very important energy reductions in almost all the components. Techniques are applied per component but the effect influences all the elements on the system and our approach allows to study this effect.

We observe also that the energy could be reduced much more with another system partitioning adding more hardware accelerators to discharge the processor and the memory utilisation.

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 of complex embedded systems.

7. 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 of **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. Philips Electronic Design & Tools Group, Philips Research, *DIESEL User Manual*, version 2.5 ed., June 2001.
3. T. Simunic, L. Benini, and G. D. Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," *Proceedings of the IEEE 36th DAC, Design Automation Conference*, pp. 867–872, 1999.
4. J. Henkel and Y. Li, "Avalanche: an environment for design space exploration and optimization of low-power embedded systems," *Transactions on VLSI Systems* **10**, pp. 454–468, 2002.
5. T. 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.
6. BullDast, "Powerchecker: An integrated environment for rtl power estimation and optimization," *Version 4.0*, http://www.bulldast.com.
7. ChipVision, "Orinoco: A high-level power estimation and optimization tool suite3," http://www.chipvision.com.
8. H. Jang, M. Kang, M. Lee, and al., "High-level system modeling and architecture exploration with systemc on a network soc: S3c2510 case study," *Proceedings of the IEEE DATE, Design Automation and Test in Europe conference*, 2004.
9. D. Hommais, F. Petrot, and I. Auge, "A toolbox to map system level communications on hw/sw architecture," 2001.
10. "Systemc open source," *SystemC Version 2.0.1 User's Guide*, http://www.systemc.org.
11. M. Caldari, M. Conti, P. Crippa, and al., "Dynamic power management in an amba-based battery-powered system," *Proceedings of the 6th IEEE ICECS, International Conference on Electronics, Circuits and Systems*, pp. 525–528, 2002.
12. A. Sinha and A. Chandrakasan, "Jouletrack - a web based tool for software energy profiling," *Proceedings of the IEEE 38th DAC, Design Automation Conference*, pp. 220–225, 2001.