A Unified Framework for Power-Aware Design of Embedded Systems*

José L. Ayala and Marisa López-Vallejo

Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid (Spain)
{jayala,marisa}@die.upm.es

Abstract. This paper presents a new tool for power estimation of caches built inside a unified framework for the design of embedded systems. The estimator, which works in parallel with the functional simulator of the system, has been designed to deal with different target architectures, providing high flexibility. The estimation is based on an improved analytical power model that provides high accuracy on the estimation. The proposed framework has been verified with benchmarks from the MiBench suite, obtaining good results in terms of accuracy and execution time.

1 Introduction

Power consumption has become a critical design issue in processor based systems. This effect is even more appreciable in mobile embedded systems, where the limited battery capacity, the portability of the devices and the packaging materials require low power consumption and energy dissipation. When designing complex high-performance low-power embedded systems, the designers have to analyze multiple hardware and software options by experimenting with different architectural and algorithmic tradeoffs. In most cases the information about power consumption is not available until the whole system has been specified and the prototyping results can be collected. Due to this limitation it is necessary to perform different iterations through the design flow until the power and performance objectives have been satisfied, increasing a key parameter, the *time-to market*. In order to optimize the design of power efficient embedded systems, big efforts have to be carried out to provide information of energy dissipation from the very early design phases.

When dealing with the processor of an embedded system, there are several parameters that have a strong relation with the performance-power tradeoff. Mainly, the target processor and the memory cache hierarchy [9]. In this paper we present the design and use of an accurate estimation tool for cache power consumption. The tool has been designed to be highly integrated in a retargetable design tool chain. With this work the most important source of power dissipation in the system is completely characterized from the very early design phases,

* This work was supported by the Spanish Ministry of Science and Technology under contract TIC2000-0583-C02-02.

and the subsequent design takes into account the power dissipation as a constraint. The proposed methodology is extensible to new architectures, providing a helpful framework for the design of power-constrained embedded systems.

The power estimation of the cache is based on an analytical model that has been improved by including the effects of the switching activity. Simulation results show how this fact can drastically impact in the final estimated energy.

The paper is structured as follows. Next, related work is reviewed. Section 3 presents the main objectives of this work while section 4 describes the power estimation tool and its basic enhancements. Finally estimation results are presented and some conclusions are drawn.

2 Related Work

In the last few years special emphasis has been put on power estimation and high-level optimization tools attending to the designers' demands. Some of these works are based on analytical power models that can predict with high accuracy the power dissipation in certain processor modules (cache, system clock, data path, etc). The tool presented in this paper uses the analytical power model for caches developed by Kamble and Ghose [7]. This analytical model can be very accurate[1] but can also exhibit big deviations due to the statistical distribution of the switching activity value of the bus, that is assumed homogeneous. This is due to the lack of exact simulation results (the exact value of the switching activity can only be known when an application runs on the target architecture and the transferred data and addresses are tracked). This limitation has been solved with the present work.

Current research on power estimation has moved to higher level and very early design phases. The aim of this shift is to accelerate the complex design flow of embedded systems and lead the designer decisions with power consumption constraints. Important research has been carried out in the area of high-level power estimation by macro-model characterization [8], instruction characterization [10] or VHDL description analysis [1]. The problems with these approaches are, respectively, the extremely high dependence with the target architecture, the lack of simulation information (therefore, a statistical homogeneous distribution is still assumed) and long simulation and design time due to the low-level description. Moreover, power simulators like Wattch [2] or SimplePower [13] achieve accurate power estimations in a short time, but they are still limited to simple and unrealistic target architectures.

Finally, compiler [6] and source code [11] estimation tools have been proposed, which partially could solve the target dependence problem. However, they are still narrowly coated to a macro-model target characterization where the module dependencies are not properly managed and the static analysis does not allow explicit calculation of statistical parameters. Summarizing, it can be said that there is no a single retargetable framework to design embedded systems with both performance and power constraints.

[1] 3% of estimation error.

3 Objectives

Current tools for designing embedded systems do not include power estimation as a design constraint. Available utilities to estimate power dissipation in the datapath or the cache are not fully integrated in the design flow, and do not consider realistic processors as target architectures. Furthermore, existing power estimators for caches do not reflect the inherent data correlation due to the lack of simulation information at instruction level; this usually conducts to estimation errors that make difficult the design and analysis of low power policies.

The goal of our research work is to provide an accurate cache power consumption estimation tool highly integrated in a retargetable design tool chain. Such unified framework integrates the processor and cache functional simulators with a power estimation tool that can provide accurate estimations for many different real target architectures. Moreover, the overflow of this power estimation must be negligible in the design time or the design phases, but it must extend the simulator output with the power results. The proposed methodology is also applicable to new architectures, providing a fast and useful way to analyze power consumption on the devised systems (i.e. *fully extensible*).

In this way, the resulting unified framework provides the following advantages for the designers of embedded systems: automatic generation of cross-design tools, parallel output of functional simulation and accurate cache power estimation, negligible learning time for the user and flexibility on the selection of the target processor. This is a very helpful environment for a designer, who can use these tools for many different applications, as it is the study of the influence of compiler optimizations. Such application will be described in section 5.2.

4 Design of the Tool

The design of this tool is based on CGEN ("Cpu tool GENerator") [4], a unified framework and toolkit for writing programs like assemblers, disassemblers and simulators. CGEN is centered around application independent descriptions of different CPUs[2] (ARM, MIPS32, PowerPC...), many of them present in current embedded systems. Our work is built over the CGEN core to generate embedded system design tools with explicit information of power dissipation in caches. As CGEN successfully does with cross-design tools, the cache power estimator can be automatically generated for the whole set of available target processors in order to free the designer from this annoying task.

4.1 Brief Description of the Analytical Model

As was previously said, in order to get an accurate estimation of cache power dissipation, the exact memory access pattern must be known to feed the analytical model of power consumption [7]. This model provides mathematical expressions for the several energy dissipation sources in the cache architecture and

[2] Plus environment, like ABI.

takes as arguments parameters of different kinds: architectural (cache size, way configuration, word line size, etc), technological (line and device input/output capacitors, etc) and statistical (switching bus activity, cache accesses, etc). Such analytical model uses expressions like the ones presented in equations 1 and 2 for the energy dissipated per read and write access, where `swf` is the switching factor parameter, `T` (tag size), S_t (status bits per block frame), `L` (line size) and `m` (number of ways in a set associative cache) are architectural parameters, C_x are technology terms, and the rest are the statistical arguments that describe the access. V_{dd} represents the source voltage and V_s is the voltage swing on the bit lines during sensing.

$$E_{bit\_r} = swf \cdot V_{dd} \cdot V_s \cdot C_{bit\_rw} \cdot N_{array\_accs} \cdot 2 \cdot m \cdot (T + S_t + 8 \cdot L) \quad (1)$$

$$E_{bit\_w} = swf \cdot V_{dd} \left[(\frac{1}{2} \cdot V_{dd} - V_s) \cdot N_{bit\_w} + (\frac{1}{2} \cdot V_{dd} + V_s) \cdot N'_{bit\_w} \right] \cdot C_{bit\_rw} \quad (2)$$

These expressions can be deduced from the general equation for the power dissipation $E = swf \cdot C \cdot V_{dd}^2$ taking into account the particular cache architecture and the statistics for the access pattern.

While the technology and architectural parameters are known from the architecture design phase, the access pattern depends on the particular data that are in use. Furthermore, to calculate the exact value of the switching activity, perfect knowledge of the transferred data and addresses is needed. Such simulation detail can only be achieved at instruction level, when the memory and register contents are also available.

The switching activity factor reflects the number of bit changes at the data or address bus divided by the bus width (bit change factor). The usual approach to deal with this unavailable term is supposing half of the bits change their value, what is far away from being true when strong/weak data or address correlation appears in the application and a reduced number of bits switches. Therefore, over and underestimation on cache energy consumption can be explained by this limitation in the analytical model. The approach presented here overcomes this past limitation by explicitly calculating the switching activity term thanks to the information provided by the functional simulator.

4.2 Description of the Power Estimation Tool

To have access information, the running code must generate a detailed memory trace (memory address accessed, data transferred, type of the access) per memory access. To get a machine independent implementation of this trace facility, we have supplied the target machine with four new virtual registers (*trace registers*) storing the trace information: `tr0` stores the memory address accessed, `tr1` is a counter of the memory accesses, `tr2` is the bit flag that indicates a read or write access, and `tr3` stores the data read/written from/to memory. The counter `tr1` is needed to track the memory access instructions and generate the required trace. These trace registers are identically defined in every target processor. In

```
(dnh h-tr "trace registers"
() ; attributes
(register WI (4))
(keyword "" ((tr0 0) (tr1 1) (tr2 2) (tr3 3)))
() ()
)

(dni (.sym ld suffix) (.str "ld" suffix)
((PIPE 0))
(.str "ld" suffix " $dr,@$sr")
(+ OP1_2 op2-op dr sr)
(sequence ()
(set (reg h-tr 3) (ext-op WI (mem mode sr)))
;data transfered
(set dr (ext-op WI (mem mode sr)))
;execution
(set (reg h-tr 1) (add (reg h-tr 1) (const 1)))
;access counter
(set (reg h-tr 0) (ext-op WI sr))
;memory address
(set (reg h-tr 2) (const 0)))
;memory read
((m32r/d (unit u-load))
(m32rx (unit u-load))))
```

Fig. 1. RTL tracing code

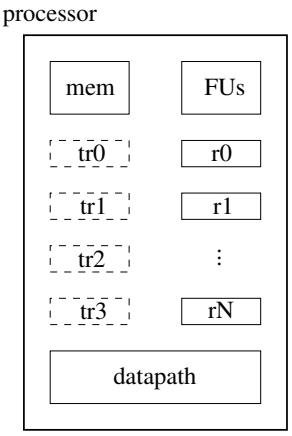

Fig. 2. Processor architecture

the same way, an identical approach is followed by the read and write operations in such trace registers. The specification of these four registers has been done in the CGEN's RTL language[3] and also the simple modifications to the processor ISA, both of them automatically accomplished by a text parser. Figure 1 shows the trace registers specification and a traced `load` instruction for a MIPS32 processor, where it can be observed how the description of the trace registers and their read and write operations are independent of the target processor (on condition that they are included in every memory access instruction). Figure 2 shows the schematic view of the processor architecture, where the virtual trace registers, used only for simulation purposes, have been represented.

The information stored in these extra registers can be read by the simulator as a normal hardware register. The simulator code in charge of reading this trace information and its analysis is also common for all the different target architectures, and has been coded in the simulator kernel. The trace information stored in these registers allows us to create a trace file which feeds the Dinero IV [3] cache simulator tool. Dinero IV is a cache simulator for memory reference traces which simulates a memory hierarchy consisting of various caches connected as one or more trees, with reference sources (the processors) at the leaves and a memory at each root, and which works in parallel with the processor simulator.

The flow of the code that performs memory tracing works as follows. For every new instruction, the trace registers are read to retrieve their information. If a new memory access happens, the trace information is stored in a file and

[3] A subset of the Scheme language.

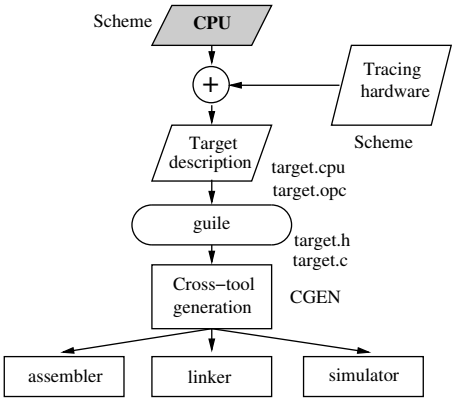

Fig. 3. Cross-tool generation

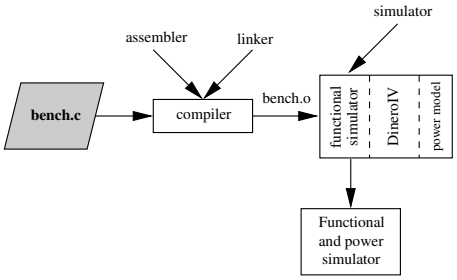

Fig. 4. Extended design flow

sent to the Dinero IV cache simulator when the size of this package of traces reaches a minimum value.

The Dinero IV functionality has been extended to perform the estimation of cache power consumption. The analytical power model has been coded in the simulator kernel and calculates the following energy expressions: energy consumption associated with the bit line precharge, energy consumption associated with the data read and write, energy dissipated in the data and address buses and energy consumed by the combinational logic (sense amplifiers, latches, address decoders...). The architectural parameters used in these expressions are based on a 0.8 μm^4 cache implementation [12]. Moreover, the statistical parameter *switching activity* has been explicitly calculated, what allows a more accurate estimation. The calculus of the switching activity parameter is obviously enabled by the detailed tracing output and allows to obtain the temporal evolution of the energy consumption with respect to the data and address correlation.

Figure 3 shows the proposed extended cross-tool generation for the design of embedded systems with explicit cache power estimation. As can be noticed, the designer must only provide the hardware description of the new target architecture which will be the core processor of the embedded system. This hardware description is automatically extended with the tracing registers and, from there on, the rest of tools are generated.

Figure 4 shows the design flow corresponding to a typical cross-design environment (host and target machine are not the same) with the particularity of concurrent functional and power simulation. In this case, the designer should only provide the source code for the benchmark, as in the common case. This source code is cross-compiled and linked and, after that, simulated for the target processor with the new tool. In this way, the simulation output has been upgraded with power consumption information.

[4] Technology available when writing this paper. It can be easily updated with new info.

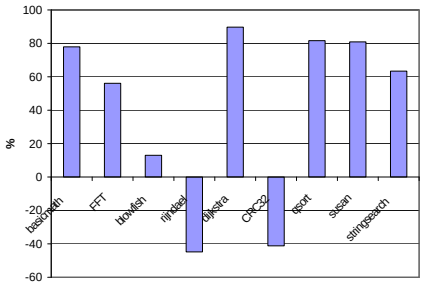

Fig. 5. Difference between average values (power consumption per write access)

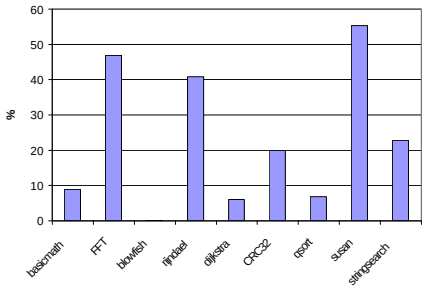

Fig. 6. Difference between standard deviations

5 Experimental Results

The proposed methodology and tools have been applied to different real benchmarks from the MiBench suite [5], compiled on an x86 machine for a MIPS32 target processor. However, any processor among MCore, i960, ARM, PowerPC or others could have been selected with similar results. The simulation time has also been analyzed resulting on a low overhead for the power estimation with respect to the functional simulation. This estimation overhead is directly related to the desired granularity (power estimation per number of simulated instructions) and to the quality of the implementation. We have obtained a simulation time with power estimation information in the same order of the functional simulation time. For instance, when simulating a medium size benchmark (*stringsearch*), the simulation time with the power estimation goes from 1.011 s to 1.400 s[5]

5.1 Switching Activity

For the whole set of benchmarks, the improvement on accuracy related to the exact calculation of the switching activity has been first analyzed.

Figure 5 shows the percentual difference between the average values of energy consumption on this data cache for the write operation. Positive values mean overestimation of the energy consumption by the homogenous model, while negative values mean underestimation of the energy consumption. It can be observed how the simplified model overestimates the power consumption in most cases because it is not able to take into account data correlations which appreciably reduce the switching factor. On the other hand, there are two cases (*rijndael* and *CRC32*) where data presented extremely low correlation and, therefore, the real switching factor is bigger than the homogeneous assumption. This underestimation gives rise to bigger power consumption than the predicted by the simplified model. This situation is specially undesirable in the design of embedded systems with power constraints, since it can cause malfunctioning of the system.

[5] Simulation time can be easily improved with parallel programming techniques.

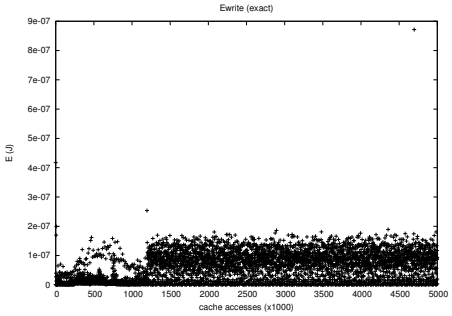

Fig. 7. Susan: exact switching factor. Write operation.

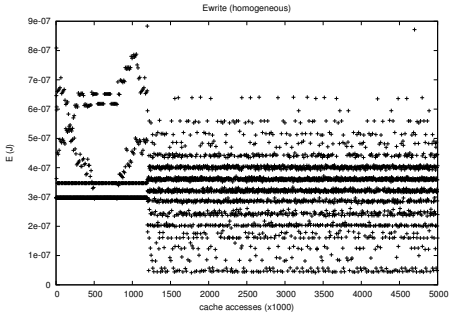

Fig. 8. Susan: homogeneous switching factor. Write operation.

Figure 6 shows the percentual difference between the standard deviation of both data distributions (exact model and homogeneously distributed model) measured for several benchmarks. This metric has been selected to analyze how spread out the power distribution is. As can be noticed, the big errors exhibited by some of the benchmarks are due to the fact that the real power distribution is far away from a homogeneous distribution.

From the whole set of experiments carried out, we have selected two benchmarks to analyze in depth. One of them is *stringsearch*, a test that searches given words in phrases, and the other one is *susan*, an image recognition package.

Now we will analyze the effect of the explicit calculation of the switching activity on energy consumption. Figures 7 and 8 show the energy bitline consumption per write access in data cache when using the calculated value of the switching activity or the homogeneous distribution, respectively, in the *susan* benchmark. The `X` axis represents every 1000 memory accesses, while the `Y` axis represents the value of the energy consumption in Joules. Both tests have been compiled without optimizations and simulated for 5 millions of memory accesses.

As can be noticed, the exact switching factor not only reflects an appreciable difference in the energy consumption per write access ($1 \cdot 10^{-7} J$ on average for the exact version, and $4 \cdot 10^{-7} J$ on average for the homogeneous version), but also a more accurate profile of the energy consumption evolution (increasing low-energy values for the earlier memory accesses, when the image file is being read from disk and written into memory). Similar behavior can be found for the *stringsearch* benchmark showing, in this case, $7 \cdot 10^{-8} J$ on average for the exact version and $3 \cdot 10^{-7} J$ on average for the homogeneous version.

The effect of the switching factor in the energy consumed per read operation is less influencing. The reduced correlation on the read accesses exhibited by the simulated applications, the linear product of the switching factor in the mathematical expressions, and the small values involved (energy per read access is much smaller than energy per write access) explain such difference.

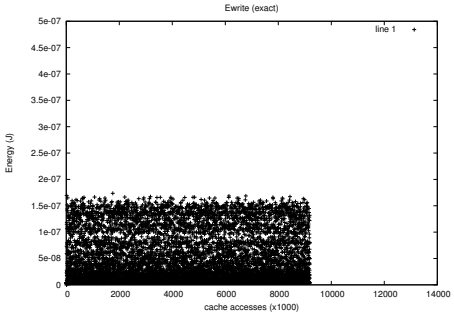

Fig. 9. Stringsearch: with compiler optimizations

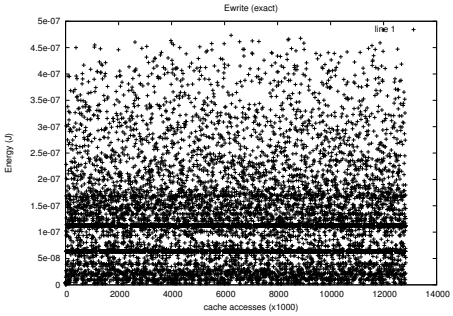

Fig. 10. Stringsearch: without compiler optimizations

5.2 Compiler Optimizations

The proposed methodology and tools can be used to evaluate energy aware compiler optimizations, next step in our research. Current performance-oriented compiler optimizations can also be characterized in terms of power awareness. For such objective, two versions of the *basicmath* benchmark have been compared, one without compiler optimizations (`-O0` option) and another one with compiler optimizations (`-O3` option). As figures 9 and 10 show, the effect of this set of compiler optimizations in power consumption per write access can result in lower power dissipation on average per instruction and, additionally, reduced number of memory accesses[6]. Both benchmarks have been simulated up to completion. We can conclude that there is a reduction on the energy consumption when compiler optimizations are applied. It is partially due to the decrease of the number of instructions and memory accesses, and it is also caused by a lower energy per instruction produced by the increased data correlation.

6 Conclusions

We have presented a unified framework of cross-design tools, which supplies a helpful environment for the efficient design of power-aware embedded systems, providing power dissipation information from the very early phases of the design flow. Furthermore, the tools, which are automatically generated, allow free selection of the main power consumers in embedded systems, i.e. target architecture and cache hierarchy configuration. Only by providing energy information in parallel with the performance simulation, the designers can lead their decisions based on both constraints without large computing overhead. Moreover, the design flow presents high flexibility since the power estimation for the cache hierarchy can be performed for different target processors.

[6] In this way, the total energy consumption is also minimized.

The design of the cache power estimation tool has been carried out using the unified environment of GNU cross-design tools. The power estimation utility bases its results on an analytical power model, whose application has been significantly improved by explicitly calculating the statistical switching activity.

The proposed methodology and tools have proved their effectiveness with real benchmarks from the MiBench test suite cross-compiled for a MIPS32 target processor, obtaining excellent results. Besides, the unified environment can also be used for many interesting applications, as it can be the power consciousness analysis of current and future compiler optimizations.

Acknowledgements The authors would like to thank José Manuel Moya for his useful and precious comments about the GNU tools.

References

1. J. M. Alcântara, A. C. C. Vieira, F Gálvez-Durand, and V. Castro. A methodology for dynamic power consumption estimation using VHDL descriptions. In *Symposium on Integrated Circuits and Systems Design*, 2002.
2. David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *International Symposium on Computer Architecture*, 2000.
3. Jan Edler and Mark D. Hill. Dinero IV, 2002. http://www.cs.wisc.edu/∼markhill/DineroIV/.
4. GNU. CGEN, 2003. http://sources.redhat.com/cgen/.
5. Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Annual Workshop on Workload Characterization*, 2001.
6. I. Kadayif, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and A. Sivasubramaniam. EAC: A compiler framework for high-level energy estimation and optimization. In *Design and Test in Europe*, 2002.
7. M. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *International Symposium on Low Power Electronics*, 1997.
8. Johann Laurent, Eric Senn, Nathalie Julien, and Eric Martin. High-level energy estimation for DSP systems. In *International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2001.
9. B. Moyer. Low-power design for embedded processors. *Proceedings of the IEEE*, 89(11), November 2001.
10. S. Nikolaidis and Th. Laopoulos. Instruction-level power consumption estimation embedded processors low-power applications. In *International Workshop on Intelligent Data Acquisition Computing Systems: Technology and Applications*, 2001.
11. E. Senn, N. Julien, and E. Martin. Power consumption estimation of a C program for data-intensive applications. In *International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2002.
12. S. E. Wilton and N. Jouppi. An enhanced access and cycle time model for on chip caches. Technical Report 93/5, DEC WRL, 1994.
13. W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of SimplePower: A cycle-accurate energy estimation tool. In *Design Automation Conference*, 2000.