Predictable Design of Low Power Systems by Pre-Implementation Estimation and Optimization

Wolfgang Nebel

Oldenburg University and
OFFIS
D-26121 Oldenburg
Tel : +49-441-9722-280
Fax : +49-441-9722-282
e-mail : nebel@offis.de

Abstract - Each year tens of billions of Dollars are wasted by the microelectronics industry because of missed deadlines and delayed design projects. These delays are partially due to design iterations many of which could have been avoided if the low level ramifications of high level design decisions, at the Architecture- and Algorithmic-level would have been known before the time consuming and tedious RT- and lower level implementation started. In this contribution we present a System-level design flow and respective EDA support tools for low power designs. We analyze the requirements for such a design technology, which shifts more responsibility to the system architect. We exemplify this approach with a design flow for low power systems. The architecture of an Algorithm-level power estimation tool will be presented together with some use cases based on an EDA product which has been commercially developed from the research results of several collaborative projects funded by the Commission of the European Community.

I Introduction

According to [1] 85% of all design projects finish late if they finish at all. The same source states that all projects are late by 53% of the originally estimated design time. Gartner/Dataquest [2] reported about the number of design iterations and the design time. From these data we can estimate the average design time for current designs to be some 10 months, the expected design time for next designs to be like 15 months. We can also estimate from this report that an average of 4.7 design iterations is needed to complete a design. Our conclusion is that NRE cost are not determined by the increasing mask cost, but rather by the design cost, secondly that the design community is obviously pessimistic with regard to the design efforts for new designs and thirdly that the design cost could be significantly reduced if design iterations could be avoided.

If we assume that the average employment cost for a design engineer in high cost regions is some US$ 200k and if we further expect an additional cost of US$ 30 k for EDA licenses, about 1/3rd of the total design cost or some US$ 70 k per design engineer are spent due to unexpected design iterations. These are typically due to late detection of design errors or because design problems are found at a very late stage of the design process. Even worse, these delays often cause missed market opportunities if competition is able to enter the market earlier and gains large market shares during the most profitable market window. A delay in market entry of six months can result in reduced revenues of up to 50%. In conclusion, delays and unnecessary design iterations cost the industry tens of billions of Dollars each year.

The reasons for the large number of design iterations are manifold: less predictable semiconductor fabrication processes, more aspects to be considered, complexity of the designs. However, there is one common issue between these reasons: The problems are detected too late! Months of tedious design time had already been spent and many CPU-hours of verification had been used to refine the design and verify certain aspects.

Here we propose a shift in the design process towards earlier phases. By trying to understand the design including non-functional properties at the earliest possible time as good as possible and by evaluation design alternatives as accurate as possible from the very beginning, it is possible to enter the RT- and lower levels of the design flow with a significantly increased level of confidence and with a much better architecture. Both benefits help to avoid later design iterations. The savings during the back end design flow and the eliminating of respins by far pay off the investments in time and EDA tools at the System-level.

This paper in particular addresses the power aspect of Systems on Chip, which is becoming one of the most limiting factors for utilizing the opportunities offered by current and future semiconductor processes. Intel's Andy Grove has stated the power problem to be the most pressing challenge and Synopsys' Art de Geus said that 11% of silicon first spins did not meet the power requirements. These statements are not surprising when high performance microprocessors consume in the order of 100 Watts and high bandwidth 3G cellular phones demand a computation performance which increases faster over 1G and 2G cellular phones than Moore's law of semiconductor technology.

II Problem Statement: Power Estimation at System-level

Due to space restrictions in this article, we limit the discussion of power dissipation to switched capacitance power. This part of the total power is still the dominating part in technologies of 90 nm and larger for high performance applications. Leakage power is gaining importance of sub 90 nm technologies and mobile applications already today. Equation 1 allows calculating the power consumption of a switched capacitor. At the Transistor-level C_{load} includes the parasitic gate overlap and fringing capacitances as well as the Miller capacity. a models the switching probability of the transistor during a cycle of the clock toggling at frequency f. V_{dd} is the supply voltage.

$$P_{swcap} = \frac{1}{2} C_{load} \cdot a \cdot V_{dd}^2 \cdot f \qquad (1)$$

This formula, however, can not be applied at the System-level directly, because the design objects at this level are not capacitors. We are rather allocating CPUs, memories, busses, custom parts, analogue subsystems etc. The design decisions are not related to low level design objects, but first to choose between algorithms, to optimize them, to partition between HW and SW, and finally to map functions to processors or to ASIC-style circuits.

Selecting the most power efficient algorithm out of a repertoire of available and functionally equivalent ones requires an estimate of the to-be-expected power consumption of an implementation of the different algorithms. Of course the comparison must be based on power efficient realizations of these algorithms without the need to really implement them.

Once an algorithm has been chosen, it can be optimized for low power. First the control flow can be optimized to reduce the number of control statements, e.g. by different kinds of loop unrolling strategies. Additionally these transformations extend the scope of local statement reordering and pave the way to local memory access optimizations [3].

Implementing an algorithm or its computational kernel by application specific hardware can significantly reduce the power consumption and relieve the processor from computation intensive tasks. This may allow a significant downgrading of the processor to a cheaper and less power consuming one.

The data of the algorithms is typically specified in terms of floating-point variables and arrays. For a hardware implementation a more efficient data representation is possible, e.g. fixed-point data types of adequate precision for the intended application. Algorithmic-level power estimation is applied to evaluate the impact of the mentioned algorithmic transformations and design decisions [4].

In order to enable the paradigm of efficiently and reliably exploring the design space at System-level, these steps need to be supported by EDA tools. The problem of power estimation at this level is that the target hardware is not designed yet. The building blocks of that hardware are not yet allocated; the control and data communication between these components is yet to be defined. Hence, before being able to predict the power consumption, it is necessary to estimate the global target architecture and its activity.

Once that is done, Equation 1 can be replaced by a more abstract interpretation as shown in Equation 2 [5]:

$$P_{dynamic} = N_a \cdot C_{avg.} \cdot V^2 \cdot f_{comp} \qquad (2)$$

N_a is the number of activations of the respective module per computation iteration, $C_{avg.}$ is the average switched capacitance of the module per activation, V the supply voltage of the component, and f_{comp} is the iteration (sampling) frequency of the algorithm. The number of modules and their activation strongly depends on the scheduling, allocation and binding, which have not yet been performed at the Algorithmic-level. To evaluate Equation 2, assumptions about the scheduling, the allocation and binding as well as the interconnect and storage architecture have to be made.

Additionally, power models for the components must be available. In case of standard components these models can be generated by simulation and power characterization based on lower level power analysis tools [6] and appropriate power models [7], [8]. Hence Algorithmic-level power analysis includes the following steps: Architecture estimation (scheduling, allocation, binding of operations and memory accesses, communication architecture estimation including wire length prediction), activation estimation, and power model evaluation. A generic power analysis and estimation flow is depicted in Figure 1.

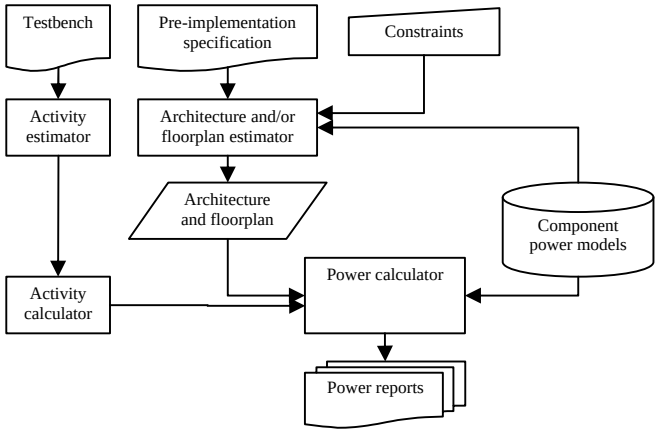

Fig. 1: Generic Power Analysis and Estimation Flow

III Algorithmic-level Power Estimation

We have identified four necessary components of an Algorithmic-Level power estimation flow:

- Architecture estimator,
- Activity estimatior,
- Library of power models for macro modules,
- Power estimation kernel,

which will be described in more detail in the following sections.

A. Architecture Estimation

The main challenge of Algorithmic-level power estimation for hardware implementations is the difficulty to predict the structural and physical properties of a yet to be designed power optimized circuit. Existing approaches to solve this problem rely on a power optimizing architectural synthesis of the design before power analysis.

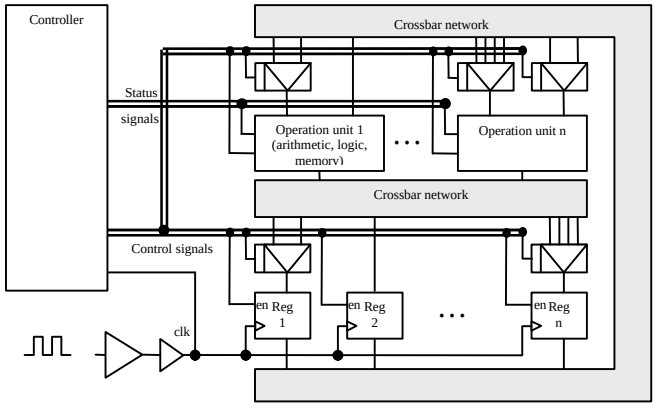

Fig. 2: Target Architecture Template

Generating an abstract architecture for low power ASIC-style implementations of computational kernels requires applying the usual steps of high-level synthesis, however, under consideration of a cost function that includes the expected power consumption. The computational kernel is typically computation intensive and hence implemented by a custom datapath with a respective controller as shown in Fig. 2.

Lets us assume that the algorithm under consideration is represented by its Control and Dataflow Graph (CDFG) [9], [10]. The task of high-level synthesis is to schedule the operations of the CDFG into control steps and to map the CDFG to a target architecture. This mapping requires allocating the necessary resources, to bind the operation nodes of the CDFG to these resources in a power-optimized way, and to develop the respective controller. In deep sub-micron technologies, however, the performance and power consumption is also dependent on the interconnect, which itself depends on the floorplan of the circuit.

Due to the strong dependence between the schedule, the resource allocation, the resource binding and the floorplan, these steps should ideally be performed in a combined optimization step. Approaches to combine several of these tasks of high-level synthesis into one optimization loop have been proposed [11], [12], [13]. The common feature of these optimization flows is to apply a set of moves on a preliminary design, to evaluate the impact of these moves, and following an optimizing heuristic like, e.g. simulated annealing, applying further moves until a stopping criterion is fulfilled.

The approach described in [13] applies moves changing the schedule and the binding. Before evaluating the cost function, they perform a floorplanning step during each iteration. Alternatively [11] use allocation and binding moves followed by a floorplanning for cost estimation, while [12] includes allocation, binding and floorplanning moves into their optimization heuristics (see Fig. 3). The upper part of the figure shows the outer loop of the optimization, during which binding and allocation moves are preformed. If, based on a preliminary power estimate, a binding/allocation move is promising, then the floorplan is updated and optimized by several floorplan moves in an inner loop, as shown in the lower part of Fig. 3. In this case the moves consist of resource allocation (sharing/splitting) and binding moves as well as floorplan related moves. The results show a significant improvement compared to interconnect unaware power optimization.

The performance of most heuristic optimization algorithms depends on the quality of the initial state before applying any of the moves. In case of Architectural-level power optimization, this initial configuration includes a schedule, a set of allocated resources, an initial binding, and a floorplan.

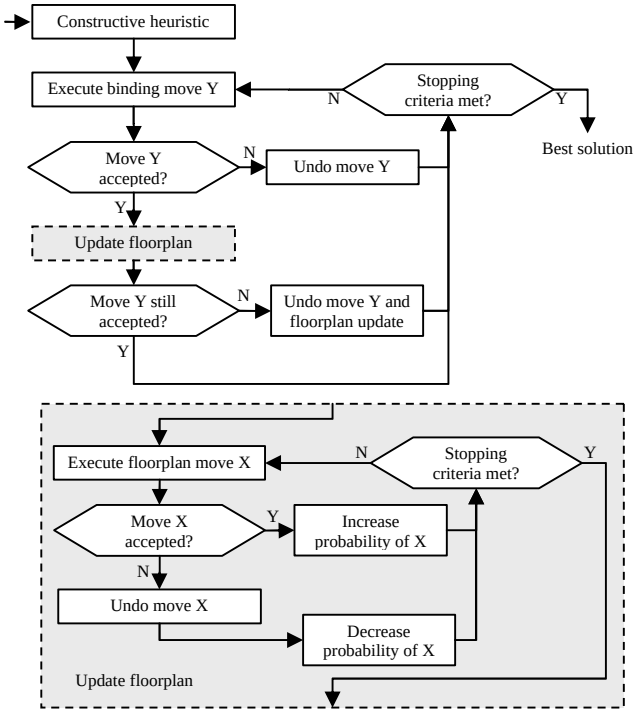

Fig. 3 Combined Power-aware Allocation, Binding, and Floorplaning

During scheduling each operation node of the CDFG is assigned to:

- exactly one control step or
- in case of chaining to an execution position within one control step or
- in case of pipelining to a sequence of control steps.

The resulting schedule defines the level of parallelism in the datapath and hence the number of required resources.

The schedule determines the usage of pipelining and chaining. While pipelining can be a means to reduce power by isolating the propagation of unnecessary signal transitions even within one operation unit, chaining causes the propagation of such glitches through several operation units in one clock cycle and hence increases the power consumption.

Power-aware scheduling avoids the allocation of resource compatible computations with low correlation of the operand values in neighboring control steps. It rather tries to allocate operations with highly correlated data into consecutive steps [14]. This strategy allows binding these operations to the same HW-resource, which will consume less power due to the low activity at its inputs.

The allocation of resources defines which and how many resources are used to implement the CDFG. The binding step assigns exactly one operation unit to each of the operations of the CDFG. Several operations can be assigned to the same operation unit (resource sharing) if they are scheduled into disjoint control steps and the operation belongs to a subset of the operations that can be implemented by the same unit.

The valid set of target units of the resource binding depends on the set of operations these units can perform. This opens further possibilities for power optimization, because more than one type of operation unit can be chosen as target unit, influencing the resulting power consumption. For example an addition can be bound to a carry-look-ahead adder, a carry-save adder or an ALU. Similarly variables and arrays can be mapped to registers or memories. Typically arrays will be mapped to memories while single variables will be mapped to registers.

The resource allocation and binding affects the power consumption of the datapath due to several effects. The power consumption of each operation unit strongly depends on the switching activity of its inputs. When processing real application data, the internal data applied to the operation units will usually not be independent, but highly correlated over a wide range of input data. Applying consecutive input data of high correlation to an operation unit reduces its power consumption. An established measure for the input switching activity is the average Hamming-Distance of a sequence of input patterns [7]. Analyzing the input streams of the operations allows assigning the operations to operation units in a power optimized way by exploiting these data correlations. Since this assignment is an NP-complete problem, different heuristics have been proposed. The approach in [15] uses an activity matrix to capture this data dependency and includes control flow information and state transition probabilities into the power analysis, while another proposal focuses on the iterative nature of data dominated designs [16].

By sequentially applying these techniques an initial schedule and architecture can be generated, which, however, is not a global optimum solution, but which can serve as a good initial solution for the simultaneous optimization described above.

So far only the architecture of the datapath and its floorplan have been estimated. The power consumption of the controller depends on its implementation, i.e. the number of registers and their activity, the implementation of the state-transition and output functions, and their signal probabilities, which are known after the architecture estimation process described above.

B. Activity Estimation

The input specification for Algorithmic-Level power estimation is typically an executable specification in terms of a programming or System-Level design language, e.g. C or SystemC. It is hence straightforward to estimate the activity of the algorithm by executing the algorithm and sampling the activity of the variables and operations of the algorithm.

This process can be automated by an automatic instrumentation of the source code. This instrumentation takes care of capturing the data streams during execution.

C. Power Models

The operation units used in the generated architecture are pre-designed and power-characterized modules, like multipliers, memories, adders, ALUs, comparators, subtractors etc. In case of standard components these models can be generated by simulation and power characterization based on lower level power analysis tools [6] and appropriate power models [7], [8]. These power models should be parameterized with respect to structural aspects, e.g. bit-width, and activity parameters. The Hamming-Distance between consecutive input vectors has proven to be a reliable parameter to capture the input activity for such modules. Alternatively higher order functions of the switching probability distribution of input signals, e.g. momentums have been applied as parameters of high-level power models for macro modules [17].

The interconnect power depends on the topology of individual wires and their activity. Hence power models for interconnect are parameterized by the wire length and the signal activity. These models need to be calibrated with respect to the placement and routing tools used as well as with the process technology. Such empirical models can include estimators for the wire topology and the number of vias [18].

D. Power Estimator

It is the task of the power estimator to provide a computational framework and user interface, which integrates the techniques presented above into an EDA tool.

In particular the Power Estimator has to provide a language front-end which allows reading algorithmic specifications in a suitable language. C is a commonly used language for high-level specifications. SystemC is gaining momentum for the specification of concurrent algorithms and system architectures. In the example tool-structure shown in Fig. 4 the front-end includes a language parser which extracts the CDFG of the algorithm and automatically instruments the source code. The System-level designer can then execute the instrumented source code with application stimuli or other representative testbenches. During execution

the values of the variables and the input and output vectors of operations are captured in an activity file. This activity can be attributed to the respective resources of the datapath and interconnect for later power calculation.

The top right hand part of Fig. 4 shows the architecture estimation. It is key that the estimated architecture is optimized for low power. Hence it needs to construct a datapath and respective controller that minimizes the switching activity. As described above, iterative optimization techniques have to be applied to generate a power efficient resource allocation, schedule, binding and floorplanning. This iterative procedure requires a feed-back from intermediate power estimates of the temporary solutions as indicated in Fig. 4.

Once a stable solution has been found, its power consumption can be presented to the designer in various views to give him a fast feed-back on the ramifications of design decisions.

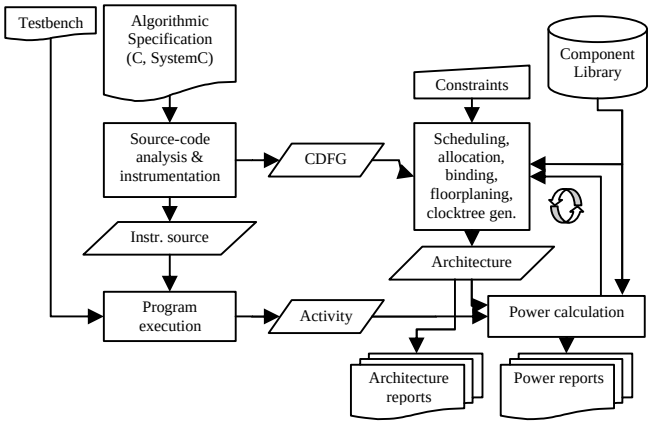

Fig. 4: Example Tool Structure of an Algorithmic-level Power Estimator

The accuracy of the power analysis depends on how well the assumed architecture matches the final architecture. This final architecture is subject to many parameters, e.g. the design style specific architecture templates, which are the main differentiating factors in times of fabless semiconductor vendors, or the tool chain applied at the later phases of the design process (RT-level synthesis, floorplanning, routing, clock tree generation). Hence an architecture estimator should either consider the design flow and style applied to the real design, or generate an abstract architecture of such high quality that it can be implemented without further global changes, however, without limiting local optimizations. In Fig. 4 the architecture output contains such a description of the architecture.

IV Tool Example

In this section we will explain how the System-level designer can utilize such early estimations. Most of the above mentioned techniques have been integrated into the ORINOCO [1] tool suite [19], which we use here for demonstration purpose.

The ORINOCO tool suite consists of two characterization tools, which generate the power models in the component library, and the power estimator ORINOCO DALE. The tool suite implements the tool structure as depicted in Fig. 4.

Fig. 5 shows the top-level hierarchy browser of the tool after estimating the power consumption of a Vocoder design.

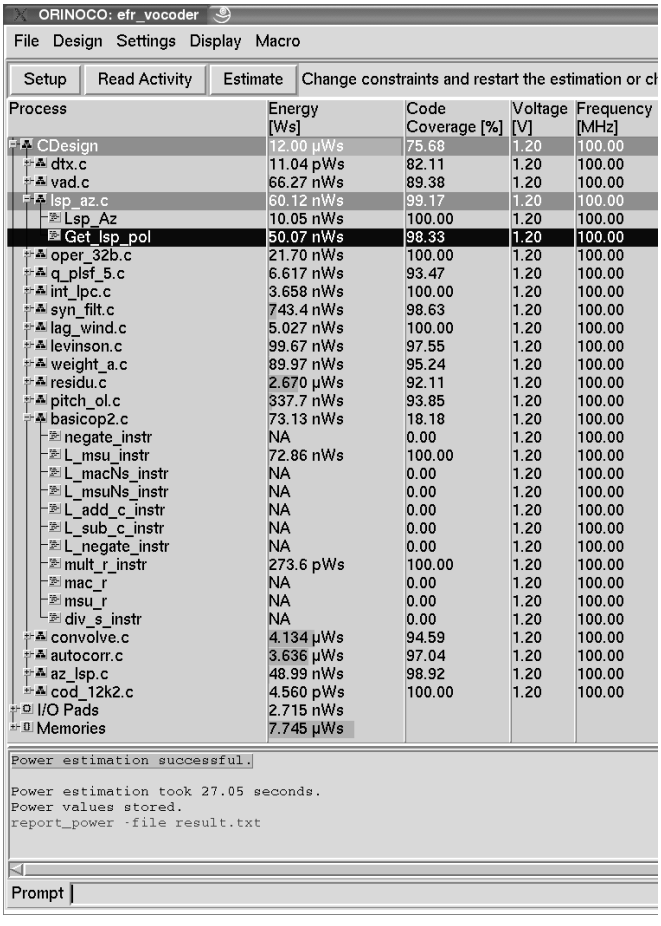

Fig. 5: Design Hierarchy of Vocoder Design

After power estimation it is easy to identify the computational kernels and hot spots of the design. The designer can now perform an in-depth analysis of the power breakdown of the most power consuming processes in the design. Fig. 6 shows an example of such a detailed power view. For each process the power is broken down with respect to the sources of the power consumption: Clock, Interconnect, Controller, Functional Units, Registers, Memories.

Additional views show e.g. the schedule, the memory accesses (Fig. 7) or the floorplan of the design. The memory access traces are of particular help when optimizing the

[1] Prototypes of the ORINOCO tools have been developed under partial funding of the Commission of the European Union within the fourth and fifth research framework as part of the research projects PEOPLE and POET. ORINOCO is now available as product from a commercial EDA vendor.

memory usage and architecture of the system, which often is the main cause of power consumption.

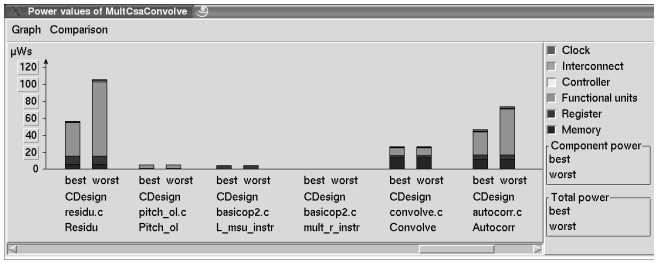

Fig. 6: Power Breakdown View

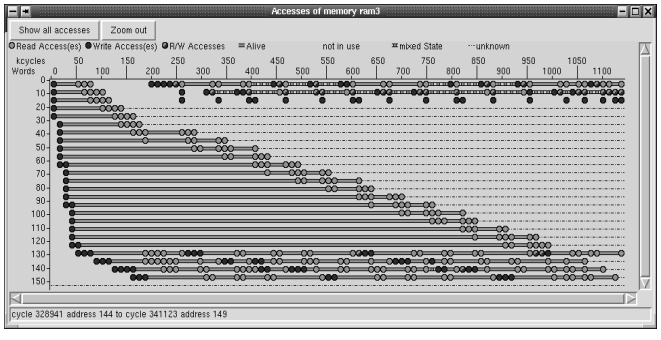

Fig. 7: Memory Access Traces

V Conclusions

With this contribution we have tried to motivate a shift in design effort towards the System-level in order to avoid unnecessary design iterations during later phases of the design. A small investment in design resources and a shift of responsibilities also for non-functional properties towards this phase will significantly reduce the design time and cost at the later design stages and in addition result in superior system architectures.

An early analysis and optimization of the algorithms to be implemented and an exploration of design alternatives requires a design technology, which enables the designer to evaluate the ramifications of high-level design decisions to the lower levels before refining the design down to these levels. It can be enabled by decision support and design tools, which predict physical properties of the design to be implemented at the earliest possible instant and give guidelines for the architectural designer.

We have illustrated this approach with a System-level design methodology for low power. The necessary steps for such an early estimation have been analyzed and requirements derived. A tool structure for an Algorithmic-level power estimation tool has been developed and use cases have been shown based on a commercial implementation of such a tool, which is the product version of a prototype having previously been developed as part of a European collaborative research project.

A more concise introduction into System-level power analysis and estimation can be found in [20].

References

[1] Jones, H., H., “Analysis of the Relationship between EDA Expenditures and Competitive Positioning of IC Vendors,” IDS, 2002.

[2] Smith, G., “ASIC Design Times Spiral out of Control”, Gartner Dataquest, 2002.

[3] Sarker, B., Nebel, W., and Schulte, M., Low Power Optimization Techniques in Overlap Add Algorithmus, in *Proc. Int. Conf. on Computer, Communication and Control Technologies: CCCT´03*, Orlando, July, August, 2003.

[4] Stammermann, A., Kruse, L., Nebel, W., Pratsch, A., Schmidt, E., Schulte, M., and Schulz, A., System Level Optimization and Design Space Exploration for Low Power, in *Proc. Int. Symposium on System Synthesis*, Montreal, September, 2001.

[5] Mehra, R., and Rabaey, J., Behavioral Level Power Estimation and Exploration, in *Proc. First Int. Workshop on Low Power Design*, Napa Valley, April, 1994.

[6] http://www.bulldast.com/powerchecker.html.

[7] Von Cölln, G., Kruse, L., Schmidt, E., Stammermann, A., and Nebel, W., Power Macro-Modelling for Firm-Macros, in *Proc. PATMOS*, Göttingen, September, 2000.

[8] Schmidt, E., von Cölln, G., Kruse, L., Theeuwen, F., and Nebel, W., Automatic Nonlinear Memory Power Modelling, in *Proc. Design, Automation and Test in Europe (DATE)*, Munich, March, 2001.

[9] Girczyc, E.F., and Knight, J.P., An ADA to Standard Cell Hardware Compiler Based on Graph Grammers and Scheduling, in *Proc. IEEE Int. Conf. on Computer Design*, October, 1984.

[10] Raghunathan, A., and Jha, N.K., Behavioral Synthesis for Low Power, in *Proc. IEEE Int. Conf. on Computer Design*, October, 1994.

[11] Zhong, L., and Jha, N.K., Interconnect-aware High-level Synthesis for Low Power, in *Proc. Conference on Computer Aided Design*, San Jose, November, 2002.

[12] Stammermann, A., Helms, D., Schulte, M., Schulz, A., and Nebel, W., Binding, Allocation and Floorplanning in Low Power High-Level Synthesis, in *Proc. ACM/IEEE Int. Conference on Computer Aided Design*, San Jose, November 2003.

[13] Prabhakaran, P, Banerjee, P., Crenshaw, J., and Sarrafzadeh, M., Simultaneous Scheduling, Binding and Floorplanning for Interconnect Power Optimization, in *Proc. VLSI Design*, Goa, January, 1999.

[14] Musoll, E. and Cortadella, J., Scheduling and Resource Binding for Low Power, in *Proc. Int. Symposium on System Synthesis*, Cannes, September, 1995.

[15] Khouri, K.S., Lakshminarayana, G., and Jha, N.K., Fast High-level Power Estimation for Control-flow Intensive Designs, in *Proc. International Symposium on Low Power Electronics and Design*, Monterey, August, 1998.

[16] Kruse, L., Schmidt, E., Jochens, G., Stammermann, A., Schulz, A., Macii, E., and Nebel, W., Estimation of Lower and Upper Bounds on the Power Consumption from Scheduled Data Flow Graphs, IEEE *Trans. On Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 1, February, 2001.

[17] GarciaOrtiz, A., Kabulepa, L., Murgan, T., Glesner, M., Moment Based Power Estimation in Very Deep Submicron Technologies, in *Proc. ACM/IEEE Int. Conf Conference on Computer Aided Design*, San Jose, November 2003.

[18] Stammermann, A., Helms, D., Schulte, M., and Nebel, W., Interconnect Driven Low Power High-Level Synthesis, in *Proc. PATMOS*, Torino, September, 2003.

[19] www.chipvision.com

[20] Nebel, W., Helms, D., High-Level Power Estimation and Analysis, to appear in Christian Piguet (ed.), *Low Power Electronics Design*, CRC-Press, 2004