Power Estimation Techniques for the Purpose of the Architectural Synthesis of Digital Signal Processing Algorithms

Frank J. Rem, Sabih H. Gerez, Jaap Smit
Department of Electrical Engineering, University of Twente, The Netherlands
Alexandre Heubi, Michael Ansorge, Fausto Pellandini
Institute of Microtechnology, University of Neuchâtel, Switzerland
E-mail: S.H.Gerez@el.utwente.nl or Alexandre.Heubi@imt.unine.ch

Abstract— **Architectural synthesis for digital signal processing (DSP) is the automatic generation of a VLSI implementation of a DSP algorithm. In this process, it is desirable to estimate the power consumption of potential solutions. The estimation should be fast and accurate. The *dual-bit-type* estimation method known from literature was taken as a basis and adapted for the goals of this research. Experimental results show that the followed approach gives useful results.**

I. Introduction

THE research reported in this paper is related to VLSI implementations of digital signal processing (DSP) algorithms using *architectural synthesis* (or *high-level synthesis*) [1–4]. Architectural synthesis is the process of translating an algorithmic description, as e.g. represented in a data-flow graph [5], into a description at the *register-transfer* level (RT level), a structural description consisting of an interconnection of functional units, memories and a controller. Traditionally, synthesis methods had the goal to minimize either the area for a given iteration period (time-constrained scheduling) or the iteration period given a set of resources (resource-constrained scheduling). In recent years, the minimization of power in VLSI design has received more and more attention (see e.g. [6–10]).

In its search for a solution that consumes minimal power, architectural synthesis will usually generate several (partial) solutions and estimate their power consumption (see e.g. [11, 12]). This paper presents power estimation techniques that can be used by an architectural synthesis system for DSP algorithms.

The paper is organized as follows. After a presentation of the hardware model used, the main principles of high-level power estimation are reviewed. Then, the "dual bit type" method on which the research of this paper is based, is introduced followed by some details specific to the hardware model. Experimental results showing the usefulness of the approach are finally presented.

Frank Rem's current affiliation: Océ Technologies B.V., Venlo, The Netherlands.

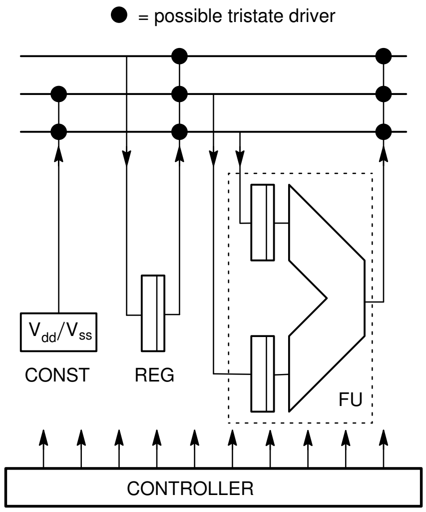

Fig. 1. The hardware model used.

II. The Hardware Model for Architectural Synthesis

When designing algorithms for architectural synthesis, one of the issues to be settled is the type of hardware that will be generated: the types of hardware units that are allowed, the constraints imposed on the way that these units may be interconnected, the number of clock phases, etc. [13]. This set of properties of the target hardware is normally called the *hardware model.*

An impression of the hardware model used for the research of this paper is given Fig. 1. The model has been kept as simple as possible in order to facilitate the coding of a simple synthesis algorithm. Note that power estimation and not synthesis was the main goal of this research. The model contains at least one functional unit (FU), a unit that performs a computation such as an adder, multiplier or ALU taken from a *module library.* Here, the module library only contains two FUs: a carry-ripple adder and an array multiplier. Both terminate their executions within a single control step (clock cycle).

FUs have registers at their inputs (so *operation chaining* is not allowed). These registers take their inputs from a private bus (no multiplexers are used), called an *input*

ProRISC Workshop on Circuits, Systems and Signal Processing, November 1997.

bus. FUs can write to any of the input buses using *tristate drivers*. Stand-alone registers are used to store intermediate values that cannot be directly passed to a next FU. These registers also have a private input bus and can write to any input bus using tristate drivers.

Constant signals used for multiplications are not stored in a memory but are *hardwired* by connecting the individual input bits of a tristate driver to either positive or negative power supply.

The type of algorithms considered for synthesis do not contain conditional constructs and can be represented by *iterative data-flow graphs* (IDFGs) [14]. Therefore, the controller does not need to receive any status signals from the data path. It cycles through a linearly ordered set of states and generates the correct *enable* signals for the registers and the tristate drivers.

The complete design is realized in *standard cells* using VHDL synthesis.

III. High-Level Power Estimation Methods

One, basically, has two requirements for a power estimation method during architectural synthesis:

- the method should be fast to compute as it will be invoked many times;
- the method should be reliable; it is important that the ranking of design alternatives based on the estimation values is the same as the ranking based on actual values; the error between estimation and actual values is of secondary importance.

Obviously, the more detail is taken into account the more accurate an estimation will be. In VLSI design, a power estimation based on circuit-level simulation will normally be more precise than a switch-level simulation. When one continues to go up in the level of abstraction through gate level and the RT level to the system level, the degree of accuracy of estimations will decrease as the modeling becomes less detailed. On the other hand, the computational effort necessary for the estimation will decrease as well.

The main source of energy consumption in an integrated circuit is the charging of parasitic capacitances [15]. Each capacitance C will require $\frac{1}{2}CV_{dd}^2$ of energy when charged from 0 V to the positive supply voltage V_{dd}. This corresponds to the situation that the signal connected to the capacitance makes a $0 \rightarrow 1$ transition.

The challenge of power estimation is to know the capacitance values in the entire circuit and the average number of times that each capacitance is charged per second for a typical input data stream without the necessity to take all circuit details and signal properties into account. The general expression for the power consumption of a circuit is:

$$P = \frac{1}{2} V_{dd}^2 f \sum_i p_i C_i \qquad (1)$$

In this equation, f is the frequency of operation, C_i is a capacitance and p_i is the probability of a power consuming signal transition on C_i in one clock period. The index i ranges over all capacitances in the circuit. The expression is valid for any degree of granularity: one can enumerate all separate capacitances, but may try to group into a single C_i value as many capacitances as possible that have the same switching probability p_i. Therefore, one can also say that the challenge of high-level power estimation is to find models that provide sufficient accuracy while keeping the number of elements in the summation of Equation 1 as small as possible. Overviews of methods for high-level power estimation can be found in recently published review papers [16–18].

Many methods for high-level power estimation make use of *macro-models*. These are functional descriptions that compute power values for specific hardware modules (functional units, memories, etc.) as a function of module parameters (e.g. word lengths) and statistical parameters related to input signal activities. The research reported in this paper is based on such a macro-modeling method: the *dual-bit-type* (DBT) method introduced by Landman and Rabaey [19]. Its principles and some modifications are explained in the next section.

High-level estimation methods should, if possible, correctly model *temporal* and *spatial* correlations of signals. When signal values in a data stream do not change too fast, the switching activity as a result of this data stream will be less than e.g. a data stream consisting of random values. The signal values are temporally correlated. It may also be that signals belonging to different data streams simultaneously change their values (because they e.g. were both generated from one original signal). These signals are spatially correlated. Estimation methods that assume a "uniform white noise" (UWN) model for all signals, such as the one described in [20], have the tendency to strongly overestimate the power consumption with respect to methods that take signal correlations into account [21]. In the UWN model that assumes random signal values, the probability that a bit will make the power-consuming transition $0 \rightarrow 1$ for subsequent signal values is 0.25 (this transition is then as likely as the three other possible transitions $0 \rightarrow 0$, $1 \rightarrow 0$, and $1 \rightarrow 1$).

IV. The Dual-Bit-Type Power Estimation Method

In this section, the main lines of the method are explained. The discussion closely follows the ideas of [19] although it has been influenced by the authors' implementation of the method in some places. The next section more concretely deals with this implementation.

The DBT method owes its name to the fact that it partitions the bits of a data word into two types: the higher-order bits (or *sign bits*) that do not change often for subsequent signal values in a data stream, and lower-order bits (or *UWN bits*) that show random behavior. It is assumed that the data words in a stream of signal values are encoded as two's-complement fixed-point integers in the range $[-1, 1)$ as is often the case in DSP. The stream is characterized by the following parameters: the number of bits m in the fractional part of the data word (the data word has $m+1$ bits in total), the average signal value μ, the variance σ^2, and the correlation of a signal value with its previous value in the stream ρ. The following expressions are used to compute two *break points* B_0 and B_1 with $m \geq B_1 \geq B_0$. The break points are bit indices (the lower the index value, the less significant the bit). B_1 limits the sign bits and is found from:

$$B_1 = m \log_2 \sigma + \log_2 \left(\sqrt{1-\rho^2} + |\rho|/8 \right)$$

B_0 limits the UWN bits and is found from:

$$B_0 = m \log_2 (|\mu| + 3\sigma)$$

A justification for these expressions is given in the appendix of [19]. As the DBT method does not give a separate treatment to the bits between B_1 and B_0, a single break point B halfway between these two position is actually used in the modeling: $B = \frac{B_0+B_1}{2}$.

The DBT method gives a special treatment to sign-bit transitions. The sign bits of two subsequent signal values can remain unchanged (positive or negative), can change from positive to negative or the other way around. These transitions can respectively be denoted by `++`, `--`, `+-`, and `-+`.

With respect to Equation 1, the DBT model first of all considers all hardware units separately (so, different units have different C_i values). Besides, multiple C_i values are considered for each hardware unit related to whether the consumption due to sign or UWN bits is computed. In the case of sign bits, the different sign-bit transitions just mentioned are each taken into account separately. Basically, the term $p_i C_i$ in Equation 1 for a hardware unit i is then written as:

$$p_i C_i = \frac{B}{m+1} C_{i,\mathrm{UWN}} + \frac{m+1-B}{m+1} \sum_{\sigma \in S_i} p_\sigma C_{i,\sigma} \qquad (2)$$

where S_i is the set of possible sign transitions for hardware unit i, and $C_{i,\mathrm{UWN}}$, $C_{i,\sigma}$ are *effective capacitances* associated with specific signal behaviors, and p_σ is the probability for sign-bit transition σ. In the case of a hardware unit with multiple inputs the value B in the equation is obtained through averaging of the B values of the individual input streams.

The constitution of the set of possible sign transitions S_i depends on the considered hardware unit i. In the case of a register that simply copies its input data to its output, for example, the four transitions `++`, `--`, `+-`, and `-+` are sufficient. A functional unit such as an adder has, however, two inputs. Besides, its output may or may not change for specific combinations of input sign transitions. In such a case, a transition is denoted by a concatenation of the two input transitions and the output transition. Example: `++/+-/++` denotes the situation in which both inputs (and by implication) the output are initially positive and where the second input switches to negative without that the sign of the output is affected.

The effective capacitance values depend, of course, on the size of hardware units. In the DBT method, an effective capacitance $C_{i,\sigma}$, $\sigma \in S_i \cup \{\mathrm{UWN}\}$, is therefore written as an inner product:

$$C_{i,\sigma} = \mathbf{C}_{i,\sigma} \cdot \mathbf{N}_i \qquad (3)$$

where $\mathbf{C}_{i,\sigma}$ is a *capacitive coefficient vector* and $\mathbf{N}_i$ is a *complexity parameter vector*. The latter vector has as many elements as is necessary to achieve sufficient modeling accuracy. Considering e.g. a $p \times q$ multiplier, it seems logical to choose $\mathbf{N}_i = \begin{bmatrix} p & q \end{bmatrix}^T$. In the research reported here, only square multipliers have been considered which resulted in a scalar complexity parameter. Experiments with alternative complexity parameter vectors for multipliers have been reported in [22].

The computational effort necessary to perform estimations according to the DBT method is distributed into three stages:

1. *Hardware characterization.* This computation only needs to be performed once at the moment that the module library is composed. The goal of this computation is to obtain all capacitive coefficient vectors for all modules and all signal transitions.
2. *IDFG simulation.* This computation should be performed for each new algorithm to be synthesized. The computation collects data streams for each edge of the IDFG that result from a typical input data stream.
3. *RT-level power estimation.* This is the only computation that should be repeated for each tentative solution investigated by the synthesis algorithm. Here the effect of merged data streams from the IDFG (as a consequence of resource sharing) is taken into account to compute the B values for each stream. These statistics are combined with the characterization data collected in the first step to derive the power consumption of the specific solution following Equations 1 and 2.

These main ideas of the DBT method are summarized in Fig. 2 (in the figure, "HLS" stands for "high-level synthesis").

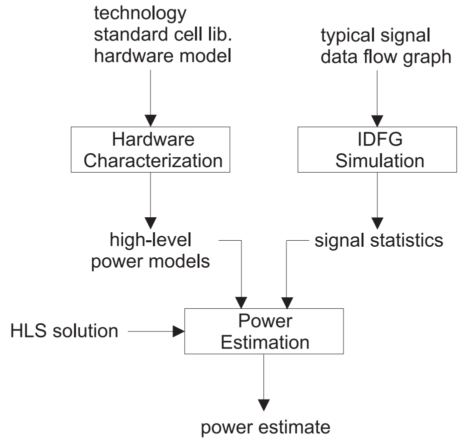

Fig. 2. The different parts of the DBT estimation method.

V. Hardware-Unit Models

In this section, all hardware units and their DBT models that are used in the hardware model (see Section II), are presented.

A. Functional Units

The IDFGs that are considered for synthesis, only have additions and multiplications as mathematical operations and, therefore, the only FUs in the synthesized hardware are adders and multipliers. Adders are characterized by a scalar complexity parameter (a single element vector $\mathbf{N}_i$ in Equation 3): the word length of its inputs as is also reported in [19].

The multiplications encountered in most digital filters concern the computation of the product of a constant and a data value. It is not accurate to consider a data stream of (possibly multiplexed) constants as a stochastic stream whose bits can be partitioned in sign and random bits. The most accurate way to deal with constants is to take them into account during characterization. It is not sufficient to repeat characterization for each constant; one should consider all possible pairs of constants in order to deal with multiplexing (resource sharing). Using the constant values during characterization, however, contradicts the principles of the DBT method that characterizes the hardware units without knowledge of the algorithms for which the units will be used.

After some experimenting the following solution was chosen (the solution is somewhat different from the one proposed by Landman [23]). Although the module library contains a single multiplier, this hardware unit was characterized in two ways. The first way models the multiplier similarly to the adder of which the bits of both input streams can be divided into sign and random bits. For the sake of simplicity, only square multipliers (with equal word length for both operands) were considered for which a scalar complexity parameter was sufficient.

The second way models the multiplier as a *gain*, a multiplier of which one input does not change while the other makes a transition. The input stream not carrying the constant determines the UWN and sign-transition regions and for both regions, a distinction is made between the constant input being positive or negative. This means that multiple effective capacitances are used for the UWN region as well (the single first term in Equation 2 becomes a summation over all possible transitions weighted by their probabilities). The transitions in the UWN region are: `nn/UU` (a negative constant on the first input and UWN on the other), `UU/nn`, `pp/UU`, and `UU/pp`. An example of a transitions in the sign-bit region is: `nn/-+` (a negative constant on the first input and a negative-to-positive sign transition on the second input). In order to obtain the effective capacitance values during characterization, random constant values are used and the characterization is repeated until the effective capacitance values have converged sufficiently. Power estimation uses the "gain model" when two subsequent data values at a multiplier's input do not change and the "multiplier model" otherwise.

B. The Register

The DBT model of the register is very simple. As this hardware unit only has one input, Equation 2 can directly be used with only four sign transitions. A scalar complexity parameter, viz. the word length is sufficient. By the way, the input registers of the functional units (see Section II) are taken into account as part of the functional-unit DBT models.

C. The Controller

A controller is by its nature not part of the data path and cannot be modeled by the DBT method. Different possibilities for controller macromodels are discussed in [24]. Here, a simple model for the controller has been used based on the parameters N_S (the number of states) and N_O (the number of output bits; remember that the controller does not have inputs). The $p_i C_i$ value of Equation 1 is computed from:

$$p_c C_c = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \end{bmatrix} \begin{bmatrix} 1 \\ N_S \\ N_O \\ N_S N_O \\ N_S^2 \\ N_O^2 \end{bmatrix} \tag{4}$$

where the subscript c for i refers to "controller". The capacitive coefficients are found by generating random controllers for a wide range of values for N_S and N_O and computing the values C_1 to C_6 that minimize the error of the outcome of Equation 4 with respect to the values obtained by gate-level simulation. The controllers were implemented using random logic.

D. Interconnections

The problem of power models for interconnections can be divided into prediction of the switching activity for each net and capacitance estimation for each net. Switching activity prediction is quite straightforward: the activity of controller output bits is exactly known and the DBT method can be used for data-path interconnections. In both cases, the only transition that matters is $0 \rightarrow 1$.

The estimation of capacitances for interconnections amount to wire length estimation. The problem of estimating wire lengths on the basis of an RT-level netlist is not trivial. Experiments based on a few high-level parameters derived from the netlist have not given sufficiently accurate results [25] and are, therefore, not discussed here. Other research on standard-cell implementations of netlists obtained by architectural synthesis also shows that the area occupied by interconnections is highly variable [26]. For more accurate results more sophisticated wire length estimation models such us used in [24] are necessary. They will be investigated as part of future research.

VI. Experimental Results

The DBT method as described above has been verified for a small IDFG example, the second-order IIR filter shown in Fig. 3. Different solutions obtained by a

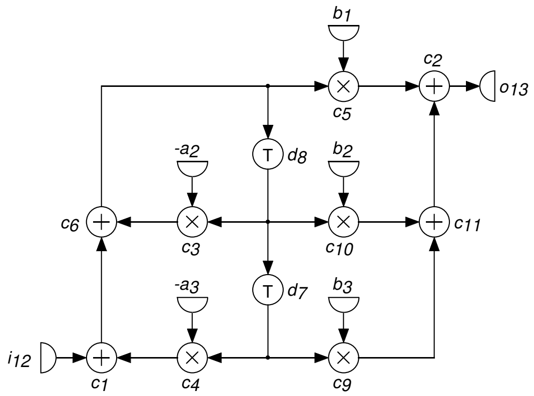

Fig. 3. The IDFG of a second-order filter section.

TABLE I
SOLUTION CHARACTERISTICS.

Name	#add	#mul	#reg	#TD	T_0	area (Mλ^2)
A	1	1	4	17	13	3.3
B	2	2	3	18	9	5.6
C	1	3	3	16	9	7.1
D	2	4	2	10	7	8.3

relatively straightforward architectural synthesis program specially written in C++ for this project [25] have been generated and analyzed. The synthesis program generates *nonoverlapped* schedules [27] and does not perform transformations such as "retiming for resource utilization" [28] (estimation rather than synthesis was the main goal of this project). The output of the program consists of synthesizable VHDL for the *Compass ASIC Synthesizer*. The different solutions were synthesized down to the layout level using the low-power `CSEL_LIB Version 3.0` standard cell library for the `alp1lv` 1μ technology provided by *E.M. Microelectronic-Marin SA*. Power simulations were performed using the gate-level power simulator provided together with `CSEL_LIB`. The power consumption of the same solutions were estimated using the DBT method using specially developed software written in C++. The results are summarized in this section; a full presentation of the results can be found in [25].

The four solutions for the IDFG benchmark example have been labeled `A`, `B`, `C`, and `D`. Table I shows the numbers of adders, multipliers, registers, and tristate drivers (TDs) used in each solution, as well as each solution's iteration period T_0 (the number of control steps per iteration) and area (in $10^6\lambda^2$). The word length of all data signals and constants was set to 16 for all solutions.

The energy per iteration in nJ for each solution as predicted by the DBT model and computed by gate-level simulation is given in Table II. Besides, the table shows the percentage of error of the estimation with respect to the gate-level simulation value. The same speech signal of 100 samples has been used for all simulations and estimations. As the interconnection estimation is not yet usable, the

TABLE II
DBT ESTIMATION EVALUATION.

Name	Gate-level value (nJ/iter.)	DBT estimation (nJ/iter.)	Error (%)
A	18.2	21.6	+19
B	16.8	19.6	+17
C	14.2	17.2	+21
D	11.7	13.4	+15

TABLE III
ENERGY WITH AND WITHOUT INTERCONNECT.

Name	With interconnect (nJ/iter.)	Without interconnect (nJ/iter.)
A	32.9	18.2
B	29.1	16.8
C	26.2	14.2
D	20.6	11.7

part of the power consumed by interconnection has been kept out of the comparison. For the sake of illustration, the gate-level simulation values including and excluding interconnections are shown in Table III. It can be seen that the energy values almost double when interconnect is taken into account.

A few observations can be made about the data in Table II. First of all, the power ranking of the solutions is correct for the DBT method as implemented here and the estimation method could be used by low-power synthesis system to select solutions with minimal power. Even when the error in estimation of 15 to 20 percent is quite high compared to the results in e.g. [21], all error values are close to each other. This systematic error can perhaps be eliminated by fine-tuning the estimation method. A third observation is that the power decreases with the increase of the parallelism in the data path (the number of multipliers increases from 1 to 4 in the solutions, while the example has 5 multiplications). This is, of course, in accordance with the expectation as fewer data streams need to be multiplexed.

More detailed results of the solution named `C` are given in Table IV. As expected, the multipliers consume most of the power. It should be noted that multiplier `mul1` is activated three times per iteration, while the multipliers `mul2` and `mul3` are only activated once. However, its power consumption is not three times but at least seven times larger than the consumption of the other two. This effect should be attributed to the multiplexing of temporarily correlated data streams which clearly cannot be neglected. The DBT method correctly deals with this effect.

VII. CONCLUSIONS

This paper has presented a power estimation approach that is suitable for architectural synthesis of DSP algorithms. The DBT method is the basis for this approach. One of the main extensions to the method is the handling

TABLE IV
DETAILED RESULTS FOR SOLUTION C.

Part	Gate-level value (nJ/iter.)	DBT estimation (nJ/iter.)
add1	0.90	0.75
mul1	10.2	12.6
mul2	1.41	1.67
mul3	1.13	1.66
reg1	0.098	0.073
reg2	0.048	0.034
reg3	0.049	0.039
controller	0.30	0.37

of multiplications with a constant. Experiments show that the estimation is quite accurate with respect to gate-level simulations and especially gives the right ranking with respect to power consumption for a set of architectural synthesis solutions. Future research should especially concentrate on accurate wire length estimations as a considerable part of the total power is dissipated in the interconnections.

Acknowledgments

Frank Rem gratefully acknowledges the *Institute for Microtechnology, Neuchâtel* for offering him the possibility to spend his practical training period and part of his graduation project period in Neuchâtel, as well as the financial support for this stay. The authors from the University of Twente would like to thank *E.M. Microelectronic-Marin S.A.* for making available the low-power standard-cell library CSEL_LIB and the associated power simulation tools.

References

[1] B.S. Haroun and M.I. Elmasry, "Architechtural synthesis for DSP silicon compilers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 4, pp. 431–447, April 1989.

[2] A. Oláh, S.H. Gerez, and S.M. Heemstra de Groot, "Scheduling and allocation for the high-level synthesis of DSP algorithms by exploitation of data transfer mobility," in *International Conference on Computer Systems and Software Engineering, CompEuro 92*, May 1992, pp. 145–150.

[3] C.Y. Wang and K.K. Parhi, "High-level DSP synthesis using concurrent transformations, scheduling and allocation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 3, pp. 274–295, March 1995.

[4] A. Heubi and F. Pellandini P. Balsiger, "An automated design methodology for the mapping of DSP algorithms into low power VLSI architectures," in *7th International Symposium on IC Technology, Systems and Applications, ISIC '97*, Singapore, September 1997.

[5] E.A. Lee and D.G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, September 1987.

[6] A.P. Chandrakasan and R.W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Boston, 1995.

[7] J.M. Rabaey and M. Pedram, Eds., *Low Power Design Methodologies*, Kluwer Academic Publishers, Boston, 1995.

[8] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, January 1996.

[9] A. Heubi, S. Grassi, M. Ansorge, and F. Pellandini, "A low power VLSI architecture with an application to adaptive algorithms for digital hearing aids," in *Signal Processing VII: Theories and Applications (Proceedings of the EUSIPCO-94 Seventh European Signal Processing Conference)*, M.J.J. Holt, C.F.N. Cowan, P.M. Grant, and W.A. Sandham, Eds., 1994, pp. 1875–1878.

[10] J. Smit, "On the energy complexity of algorithms realized in CMOS," in *ProRISC/IEEE Workshop on Circuits, Systems and Signal Processing*, Mierlo, The Netherlands, November 1996, pp. 331–341.

[11] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R.W. Brodersen, "Optimizing power using transformations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12–31, January 1995.

[12] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design and Test of Computers*, pp. 70–84, Fall 1995.

[13] D.D. Gajski, N.D. Dutt, A.C.H. Wu, and S.Y.L. Lin, *High-Level Synthesis, Introduction to Chip and System Design*, Kluwer Academic Publishers, Boston, 1992.

[14] S.M. Heemstra de Groot, S.H. Gerez, and O.E. Herrmann, "Range-chart-guided iterative data-flow-graph scheduling," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, pp. 351–364, May 1992.

[15] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[16] F.N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, December 1994.

[17] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation and optimization," in *34th Design Automation Conference*, 1997, pp. 504–511.

[18] M. Pedram, "Advanced power estimation techniques," in *Low Power Design in Deep Submicron Technology*, W. Nebel and J. Mermet, Eds. Kluwer Academic Publishers, Dordrecht, 1997.

[19] P.E. Landman and J.M. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Transactions of Very Large Scale Integration Systems*, vol. 3, no. 2, pp. 173–187, June 1995.

[20] S.R. Powell and P.M. Chau, "Estimating power dissipation of VLSI signal processing chips: The PFA technique," in *VLSI Signal Processing IV*, S. Moscovitz, K. Yao, and R. Jain, Eds., 1991, pp. 250–259.

[21] P.E. Landman and J.M. Rabaey, "Power estimation for high level synthesis," in *The European Conference on Design Automation with the European Event on ASIC Design, EDAC/EUROASIC '93*, 1993, pp. 361–366.

[22] M.E. Petersen, "Power estimation for high level synthesis," M.S. thesis, Department of Computer Science, Technical University of Denmark, Lyngby, August 1995.

[23] P.E. Landman, "Private communication," September, 1996.

[24] P.E. Landman and J.M. Rabaey, "Activity-sensitive architectural power analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 571–587, June 1996.

[25] F.J. Rem, "Power estimation for high-level synthesis," M.S. thesis, University of Twente, Department of Electrical Engineering, March 1997, EL-BSC-010N97.

[26] N.J.M. van der Eng, "From high-level synthesis to silicon," M.S. thesis, University of Twente, Department of Electrical Engineering, June 1997, EL-BSC-041N97.

[27] K.K. Parhi and D.G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Transactions on Computers*, vol. 40, no. 2, pp. 178–195, February 1991.

[28] J.M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast-prototyping of datapath-intensive architechtures," *IEEE Design and Test of Computers*, pp. 40–51, June 1991.