Implicit FSM Decomposition Applied to Low-Power Design

José C. Monteiro, *Member, IEEE* and Arlindo L. Oliveira, *Member, IEEE*

***Abstract*—Clock-gating techniques are very effective in the reduction of the switching activity in sequential logic circuits. In this paper, we describe a clock-gating technique based on finite-state machine (FSM) decomposition. The approach is based on the computation of two sub-FSMs that together have the same functionality as the original FSM. For all the transitions within one sub-FSM, the clock for the other sub-FSM is disabled. To minimize the average switching activity, we search for a small cluster of states with high stationary state probability and use it to create the small sub-FSM. Explicit manipulation of the state transition graph requires time and space exponential on the number of registers in the circuit, thereby restricting the applicability of explicit methods to relatively small circuits. The approach we propose is based on a method that implicitly performs the FSM decomposition. Using this technique, the FSM decomposition is performed by direct manipulation of the circuit. We provide a set of experiments that show that power consumption can be substantially reduced, in some cases by more than 70%.**

***Index Terms*—Clock grating, finite-state machine (FSM), implicit decomposition, low power.**

I. Introduction

POWER consumption has become a major design parameter in the project of integrated circuits, given the requirements for low power consumption in portable devices and the heat dissipation problem raised by ever higher clock frequencies and circuit density.

In this paper, we address the problem of optimizing logic-level sequential circuits for low power. This problem has received some attention recently. Several techniques for state assignment have been presented that aim at reducing the average switching activity of the present state lines, and consequently of the internal nodes in the combinational logic block (see, for example, [7], [11], and [14]). Retiming has also been tailored so that the distribution of the registers within the logic block minimizes the total amount of glitching in the sequential circuit [9].

Techniques based on disabling the input/state registers when some input conditions are met have been proposed and shown to be among the most effective in reducing the overall switching activity in sequential circuits [1], [3], [4], [13]. The disabling of the input/state registers is decided on a clock-cycle basis and can be done either by using a register load-enable signal or by gating the clock. This class of techniques is sometimes referred to as *logic level* or *dynamic power management*.

Manuscript received November 4, 2000; revised February 26, 2002.

J. C. Monteiro is with the Department of Informatics, IST, Lisbon Technical University, 1000 Lisboa, Portugal, and INESC-ID (e-mail: jcm@inesc-id.pt).

A. L. Oliveira is with the Department of Informatics, IST, Lisbon Technical University, 1000 Lisboa, Portugal, INESC-ID, and Cadence European Laboratories (e-mail: aml@inesc-id.pt).

Digital Object Identifier 10.1109/TVLSI.2002.801611

The decomposition approach we propose in this paper falls into this class of techniques. We use finite-state machine (FSM) decomposition to obtain the conditions for which a significant part of the registers in the circuit can be disabled. The original FSM is divided into two sub-FSMs, where one of them is significantly smaller than the other. Except for transitions that involve going from one state in one submachine to a state in the other, only one of the submachines needs to be clocked. To minimize the average switching activity, a small cluster of states with high stationary state probability is selected to be in the small sub-FSM. The objective is to obtain a small sub-FSM, active most of the time, that disables a much larger circuit (the large sub-FSM) and leads to potentially significant power savings.

The decomposition is performed implicitly by manipulating the original circuit, thus creating the desired submachines and their interconnections without ever needing an explicit or implicit representation of the state transition graph (STG). Although there exist binary decision diagram based techniques that can represent very large STGs implicitly, they are still limited by the size of the required data structures. Since decomposition is performed by direct manipulation of the circuit, our approach does not suffer from this limitation.

II. Related Work

Other methods have been proposed to achieve reductions in power dissipation that use a similar approach. One of the first logic-level shutdown methods is *precomputation* [1]. In this method, a simple combinational circuit (the precomputation logic) is added to the original circuit. Under certain input conditions, the precomputation logic disables the loading of all or a subset of the input registers, thereby reducing switching activity in the circuit.

Another proposal, the *gated-clock finite-state machines* approach [3], is based on identifying self-loops in a Moore FSM. If the FSM enters a state with a self-loop, the clock is turned off. In this situation, the inputs to the combinational logic block do not switch, and thus we have virtually zero power dissipation in that block. When the input values cause the FSM to make a state transition, the clock signal is again enabled and the circuit resumes normal operation.

The method for FSM decomposition that we describe in this paper can be seen as an extension of the gated-clock FSM approach. In FSM decomposition, we can consider the cluster of states that we select for the small sub-FSM as a "superstate" and then transitions between states in this cluster ares no more than self-loops in this superstate. The decision as to what states make up the "superblock" basically gives the opportunity to maximize the number of self-loops.

The idea of using FSM decomposition to achieve reduced power dissipation has also been recently proposed by other authors [5]. The idea is similar in that it is also based on the decomposition of the original FSM into two interacting finite-state machines. There are, however, two significant differences between the architecture proposed in that work and the one we propose. The first difference is that the decomposition is performed by explicitly manipulating the STG. The ability to avoid the explicit manipulation of the state transition graph enables our method to handle much larger circuits, since the number of states in this graph is, in general, exponential on the size of the circuit. The second difference is that, in this architecture, the registers are shared between both submachines, thereby not following closely the standard general decomposition topology. In our method, an additional power reduction is achieved by using additional signals that permit one submachine to disable the registers of the other when the next transition will be between states in that submachine.

A different technique that can be regarded as performing a FSM decomposition and that does not require an explicit representation of the STG was proposed in [2], thus making it close to the work we present in this paper. However, the partitioning is not based on state selection, but on the selection of nodes in the combinational logic of the original circuit. Although both methods may yield significant savings in power, the relative merits of the methods will vary from circuit to circuit, making them both useful in practice.

III. Decomposition of Finite-State Machines for Low Power

The decomposition of finite-state machines has been addressed by a number of authors, and several decomposition strategies have been proposed [6].

A. General FSM Decomposition

The more general form of decomposition breaks the state transition graph into two separate STGs and decomposes the original FSM into two communicating FSMs. Although there are several ways in which the state transition graphs of each machine can be built, the following approach is simple and intuitive and will be used in this paper.

1) Select a subset of the states in the original STG to belong to the first submachine, and let the remaining states belong to the second machine.
2) Generate two STGs, one for each submachine. Create a new state, the RESET state, in each of the two new STGs.
3) All transitions entirely inside each of these STGs are copied unmodified from the original STG.
4) Transitions between a state in the first sub-STG and a state in the second sub-STG are replaced by two transitions: one to the RESET state in the first submachine and one from the RESET state in the second submachine. The reverse is true for the symmetric case.

For a detailed description of this partition procedure and an example of the STG manipulation performed, the reader is referred to [10], where an explicit algorithm for this decomposition is presented.

B. Targeting the Partition Strategy for Low Power

Consider the decomposition procedure described above. For any transition that takes place between two states other than the RESET state in the same factor machine, only this machine needs to be active. This means that, for these transitions, we can disable entirely the other machine, avoiding all the power dissipation it incurs. On the other hand, transitions that involve the RESET state and another state are active simultaneously in both machines, because when one machine is leaving the RESET state, the other one is entering it. These transitions tend to generate the largest power dissipation because both machines are active at once.

The potential for the gains in power dissipation obtainable from this decomposition technique is larger if one selects a partition that exhibits the following characteristics.

1) One of the machines (the *small* machine) has a state transition diagram with a small number of states, and is therefore simple and dissipates a small amount of power.
2) The sum of the transition probabilities between two states in the *small* machine other than the RESET state is as large as possible.
3) The sum of the transition probabilities involving the RESET state in the *small* machine is as small as possible.

Given the general objectives described above, we use the Kernighan–Lin graph partition algorithm [8] to select a partition of the original set of states Q into two subsets Q_s and Q_l that maximize the following objective function:

$$F = \sum_{q_i, q_j \in Q_s} P(e_{ij}) - \beta \sum_{q_i \in Q_s, q_j \in Q_l} P(e_{ij}) + P(e_{ji}). \quad (1)$$

The first summation represents the total probability of the transitions occurring entirely inside the STG of the *small* machine, while the second term represents the total probability of the transitions occurring between the two machines. The coefficient β models the relative weights given to the two factors present in the equation. Empirically, we found that values between 0.5 and 1.0 for β worked best, and we selected a value of 0.7 for all experiments.

IV. FSM Decomposition Using Direct Circuit Manipulation

We now describe how the steady-state transition probabilities can be approximately computed and how the actual decomposition is performed by direct manipulation of the original network, avoiding an explicit extraction of the state transition graph.

A. State Selection

For machines with small state transition graphs, it is possible to obtain the stationary transition probabilities by solving the Chapman-Kolmogorov equations for this discrete-time discrete-transition system. However, if the state transition graph is too large to be extracted, it is not possible to compute these probabilities. We can use the fact that, for our approach, we are only interested in the transition probabilities that add up to a significant amount. In fact, one would like to select a set of states such that the sum of the transition probabilities between them is high

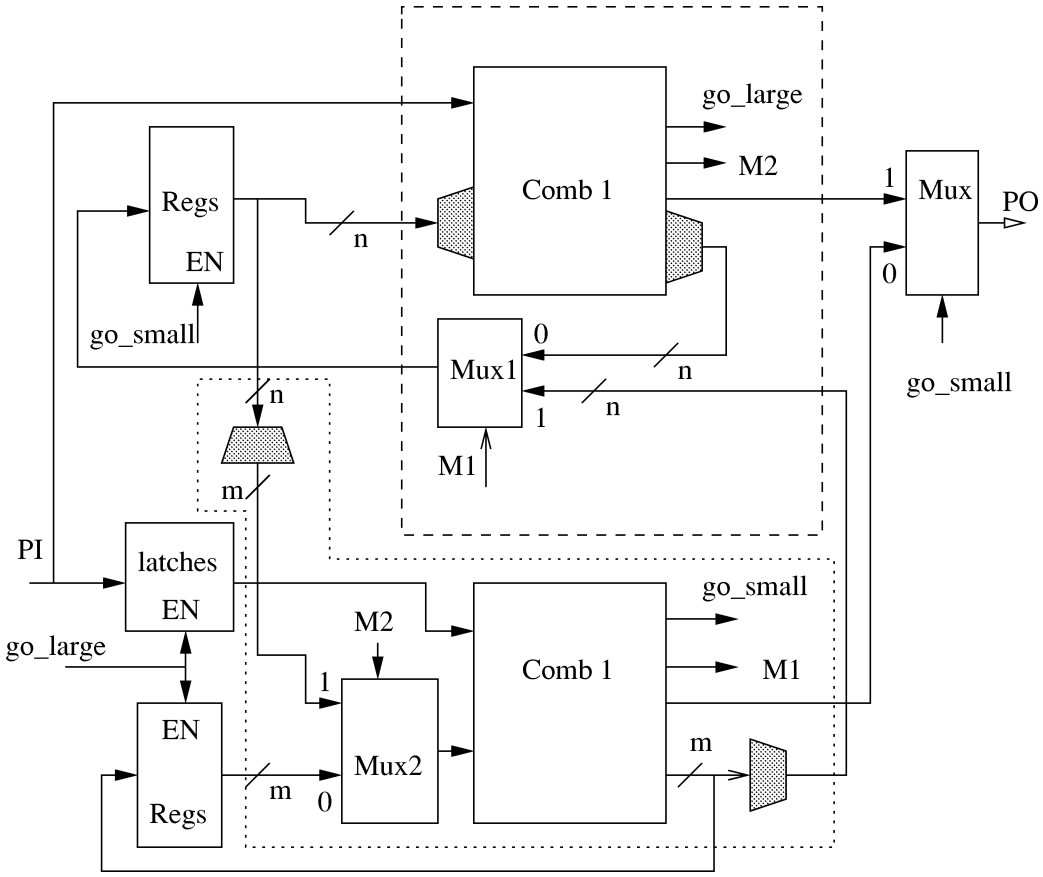

Fig. 1. Implicit FSM decomposition. The block *COMB1* represents the combinational logic of the original FSM, while the shaded blocks represent transcoders that translate between the encodings used in the original and small machine.

enough to justify the overhead incurred by our factorization procedure.

It is therefore possible to use Monte Carlo simulation to compute approximate state transition probabilities, and to use this approximation in the partition algorithm. If there are clusters of states with very high stationary transition probabilities, this type of simulation is very likely to identify them. The procedure may fail if the diameter of the state transition graph is very large, but, in this case, it is likely that no good decomposition exists. In general, this approximation works well for the cases where a cluster of interest exists. The selection of the states in the two FSMs is performed by applying the Kernighan–Lin algorithm using the estimated transition probabilities.

B. Constructing the Decomposed FSM

After the state selection is obtained, we decompose the original FSM into two communicating FSMs, as shown in Fig. 1. We will refer to each of the submachines as the *large* and the *small* sub-FSM. The large contains all the complex functionality of the original machine, while the small will only implement the functionality that corresponds to a small set of states. Assume that the large sub-FSM has m state lines and the small sub-FSM has n state lines.

The circuit is controlled by four control signals. The waveforms of these control signals are shown in Fig. 2.

Signals *go_large* and *go_small* are the enables of the small and large sub-FSMs, respectively. Signal *M2* is active only during a transition that is transferring control from the small to the large sub-FSM. Signal M1 performs the same role for the symmetric condition. To avoid combinational loops, these signals are generated in a slightly different way from each other.

The control signal *go_large* is used to disable both the registers in the large sub-FSM and a set of latches that keep transitions from the primary inputs from propagating into the combinational logic.

Signal *go_small* disables only the registers in the small machine. Disabling the primary inputs is possible, but uninteresting, since this machine will be working most of the time, at least when a good partition of the states can be found.

The shaded blocks in Fig. 1 represent transcoders that convert between the encodings used in each of the machines. Note that these blocks are relatively simple, since Q_s contains only a small number of states. By composing a *small-to-large* and a *large-to-small* transcoder with the original combinational block, it is possible to use the original logic block *Comb1* to generate both the outputs and the next state logic of the small sub-FSM, as shown at the top of Fig. 1.

The decomposition procedure described above actually solves directly the problem of reencoding and resynthesis of the decomposed machine. Yet, the decomposition performed as shown above would be uninteresting if the combinational logic of the small sub-FSM could not be reduced. Under those conditions, no power savings could take place since the small

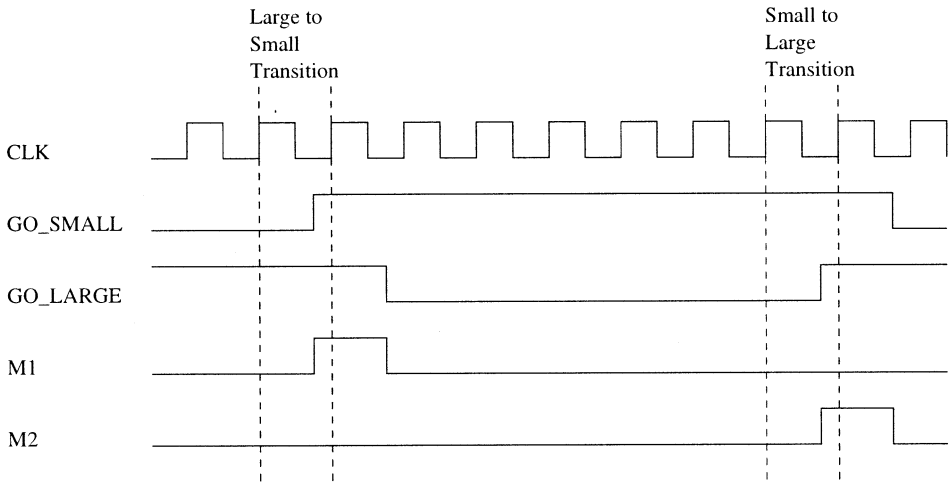

Fig. 2. Waveforms for the four control signals used in the decomposition procedure.

sub-FSM will dissipate at least as much power as the large sub-FSM.

However, note that the small sub-FSM has a much smaller number of states than the original one. Consider the logic block at the top of Fig. 1, made of a *small-to-large* transcoder, block *COMB1*, and a *large-to-small* transcoder. If k states are selected for the small sub-FSM, only $k+1$ state combinations are possible at its inputs. Using the degrees of freedom permitted by the use of controllability "don't cares," it is possible, in most cases, to reduce this circuit to a fraction of its original size, and thus obtain an implementation of the small sub-FSM that is compact and power efficient.

The state encoding for the small sub-FSM is obtained by using a simple heuristic that tries to minimize the number of bits that commute for transitions with very high probability.

V. Experimental Results

We have applied the FSM decomposition approach proposed in Section IV to circuits from the MCNC91 and ISCAS89 benchmark sets. We have selected a set of circuits for which these techniques work particularly well. Note that our claim is not that the decomposition approach we are proposing is applicable to all FSMs, but that large power savings can be achieved for many. All the results were obtained with SIS [12] on a PC running Linux, at 350 MHz, with 128 MB of main memory.

Table I shows some statistics for the circuits for which we present results. All circuits were initially optimized with `script.algebraic` and mapped to the `msu` library. The name, number of primary inputs, number of primary outputs, number of registers, and number of reachable states for each of the circuits used is given in the first four columns. For circuits `s635` and `s35932`, it was impossible to determine the number of reachable states, since the STG is too large to be extracted with the computational resources used. We also give the number of literals in column five, which is a good measure of circuit area. In the tables, the circuits are ordered by the number of registers in the original circuit. The average power dissipation (in μW) is shown in the last column, assuming a supply voltage of 5 V, a clock frequency of 20 MHz, and a general-delay model.

TABLE I
Statistics for the Circuits Used

Circuit	In	Out	Regs	States	Lts	P (μW)
cse	7	7	4	16	261	762
mark1	5	14	4	13	129	815
opus	5	6	4	10	121	803
s820	18	19	5	25	420	1194
s832	18	19	5	25	448	1220
styr	9	10	5	30	663	1612
s1488	8	19	6	48	955	2132
s1494	8	19	6	48	965	2150
s386	7	7	6	13	182	934
scf	27	56	7	115	1073	1924
s499	1	22	22	22	333	2038
s967	16	23	29	549	603	3431
s635	2	1	32	-	334	2714
s35932	35	320	1728	-	15351	186030

The power estimates were obtained using random simulation with 100 000 input vectors. To take into account the impact of increased circuit area on the switched capacitance, and since at this level of abstraction no detailed routing information is available, we used a standard wire load model [15] in the computation of line capacitance. In this manner, we take into account the impact of the increase in area of the decomposed circuit in the overall power consumption.

The results obtained using the implicit decomposition approach described here are shown in Table II. To collect the state transition statistics, we run in parallel 31 simulations (fitting in a long integer) with a sequence of 1000 input vectors. We used randomly generated input vectors simply because we did not have information about these circuits. User-specified inputs could have been used as easily. The length of the simulation run is actually conservative. As we expected, if a good cluster of

TABLE II
POWER REDUCTION AFTER DECOMPOSITION

Circuit	k	Visit	Lts	%	P (μW)	%	CPU (s)
cse	6	11	506	93	705	7.5	1.6
mark1	4	13	291	125	807	1.0	0.9
opus	6	10	350	189	623	22.4	1.1
s820	4	9	644	53	828	30.7	1.6
s832	4	9	673	50	838	31.3	1.7
styr	4	19	916	38	1570	2.6	2.2
s1488	8	25	1367	43	1144	46.3	3.1
s1494	8	25	1378	42	1350	37.2	3.1
s386	6	10	432	137	913	2.2	1.2
scf	6	33	1517	41	1597	17.0	4.1
s499	4	22	621	86	2068	-1.5	1.8
s967	8	479	1072	77	3085	10.1	5.1
s635	5	9	637	90	616	77.3	2.0
s35932	8	6347	27382	78	148249	20.3	1.3h

states exists, a shorter simulation (e.g., 100 vectors) will be able to identify it. Yet, the simulation process is very fast, finishing the 1000-vector run in less than 2 min even for the largest circuit in the benchmark set. On the other hand, we have experimented with longer runs and found no improvement on the best cluster found for any of the circuits.

These results correspond to the value of k, the number of states in the small machine, that leads to maximum power savings for each machine. To select these k states, the cost function given in (1) was used with $\beta = 0.7$, which we have found empirically to give best results.

Under column *Visit* in Table II, we give the states that were visited during the simulation of the circuit. We have observed that this number is a good indicator of the existence of a good decomposition for low power. If this number is low, then there is a high probability that the FSM stays in a small number of states most of the time. We just need to identify these states and extract them to the small submachine. If this number is close to the total number of simulated vectors, it is likely that no such cluster exists. The extreme case is when for each input vector a different state is visited, meaning that the decomposition technique we propose in this paper can never be effective. This was the case for circuits s3271, s5378, s6669, s13207, s15850, and s38417 of the ISCAS benchmark set, where 31 000 states were visited in the simulation process (some other circuits in this set were also close to 31 000). However, note that for the case of circuit s35932, the number of states visited is high and significant power savings are still possible.

Again, Table II gives the number of literals and percentage increase as well as power dissipation and percentage savings of the circuit after decomposition. As we can observe from the table, we obtain significant power savings for most circuits.

The impact on the circuit performance is relatively small. The maximum delay of the decomposed circuit is determined by the large submachine. Recall that we do not attempt any optimization for this submachine, so it is still just like the original circuit except that we have introduced a block of multiplexors to set the correct state when there is a transition from a state in the small to a state in the large submachine.

Finally, note that we were able to run the technique on all the circuits in the ISCAS benchmark set (even if a good decomposition for low power was not found) in a reasonable amount of CPU time. The run time for the smaller circuits is very small, most run in just a few seconds. The largest circuit in the set took 1.3 h of CPU time. This is in sharp contrast with the results reported in [5], where the larger circuit for which results are reported has only 48 states. This method, however, achieves higher power savings with lower area overheads.

VI. CONCLUSIONS AND FUTURE WORK

This work presents a power reduction methodology based on an FSM decomposition technique that uses direct circuit manipulation. Using this methodology, it is possible to obtain a state machine that, in many cases, exhibits a much smaller power dissipation than the original. This is achieved by selecting a small subset of the STG with high stationary probability that implements the required computations most of the time. After decomposition, register-disabling signals are added to the decomposed circuit so that the overall switching activity is minimized. The results show that power savings of over 70% are possible in some of the examples tested.

An interesting direction for future research is on the automatic selection of the size of the *small* machine. Although, in some cases, significant gains can be obtained with a variety of sizes for this machine, in other cases the result depends strongly on the adequate selection of the value of this parameter.

REFERENCES

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 426–436, Dec. 1994.
[2] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino, "Computational kernels and their application to sequential power optimization," in *Proc. 35th Design Automation Conf.*, June 1998, pp. 764–769.
[3] L. Benini, P. Siegel, and G. De Micheli, "Automatic synthesis of low-power gated-clock finite-state machines," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 630–643, June 1996.
[4] L. Benini, P. Vuillod, C. Coelho, and G. De Micheli, "Synthesis of low-power partially-clocked systems from high-level specifications," presented at the 9th Int. Symp. System Synthesis, Nov. 1996.
[5] S.-H. Chow, Y.-C. Ho, and T. Hwang, "Low power realization of finite state machines—A decomposition approach," *ACM Trans. Design Automat. Electron. Syst.*, vol. 1, no. 3, pp. 315–340, July 1996.
[6] S. Devadas and A. Newton, "Decomposition and factorization of sequential finite state machines," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1206–1217, Nov. 1989.
[7] G. Hachtel, M. Hermida, A. Pardo, M. Poncino, and F. Somenzi, "Re-encoding sequential circuits to reduce power dissipation," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 70–73.
[8] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, pp. 291–307, Feb. 1970.
[9] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 398–402.
[10] J. Monteiro and A. Oliveira, "Finite state machine decomposition for low power," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998, pp. 758–763.

[11] K. Roy and S. Prasad, "Circuit activity based logic synthesis for low power reliable operations," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 503–513, Dec. 1993.
[12] E. Sentovich *et al.*, *SIS: A System for Sequential Circuit Synthesis*. Berkeley: Univ. of California, 1992.
[13] V. Tiwari, P. Ashar, and S. Malik, "Guarded evaluation: Pushing power management to logic synthesis/design," in *Proc. Int. Symp. Low Power Electronics and Design*, April 1995, pp. 221–226.
[14] C.-Y. Tsui, M. Pedram, C. A. Chen, and A. Despain, "Low power state assignment targeting two- and multi-level logic implementations," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 82–87.
[15] D. E. Wallace and M. S. Chandrasekhar, "High-level delay estimation for technology-independent logic equations," in *Proc. ACM/IEEE Int. Conf. Computer Aided Design*, 1990, pp. 188–193.

José C. Monteiro (S'92–M'96), photograph and biography not available at the time of publication.

Arlindo L. Oliveira (S'85–M'95), photograph and biography not available at the time of publication.