Finite State Machine Decomposition For Low Power

José C. Monteiro
IST-INESC
Rua Alves Redol, 9
1000 Lisboa, Portugal
jcm@inesc.pt

Arlindo L. Oliveira
Cadence European Labs/IST-INESC
Rua Alves Redol, 9
1000 Lisboa, Portugal
aml@inesc.pt

Clock-gating techniques have been shown to be very effective in the reduction of the switching activity in sequential logic circuits. In this paper we describe a new clock-gating technique based on finite state machine (FSM) decomposition. We compute 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. This way we will have a small amount of logic that is active most of the time, during which is disabling a much larger circuit, the other sub-FSM.

We provide a set of experimental results that show that power consumption can be substantially reduced, in some cases up to 80%.

I. Introduction

Power consumption has become a major design parameter in the project of integrated circuits. Two independent factors have contributed for this. On one hand, low power consumption is essential to achieve longer autonomy for portable devices. On the other hand, increasingly higher circuit density and higher clock frequencies are creating heat dissipation problems, which in turn raise reliability concerns and lead to more expensive packaging.

In static CMOS circuits, the probabilistic switching activity of nodes in the circuit is a good measure of the average power dissipation of the circuit. Methods that can efficiently compute the average switching activity, and thus power dissipation, in CMOS combinational [10] and sequential [13] circuits have been developed.

In this work, we are concerned with 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 which 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 [12]). 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], [2], [3], [5]. 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. A common feature of these methods is the addition of extra circuitry that is able to identify input conditions for which some or all of the input/state registers can be disabled. In this situation there will be zero switching activity in the logic driven by input signals coming from the disabled registers. This class of techniques is sometimes referred to as *dynamic power management*.

The method 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 sub-machine to a state in the other, only one of the sub-machines needs to be clocked. By selecting for the small sub-FSM a cluster of states in which the original FSM has a high probability of being in, most of the time we will be disabling all the state registers in the larger sub-FSM.

The overhead associated with the FSM decomposition makes this method not very effective for typical FSMs with a small number of states. However, for large machines, impressive gains up to 80% power reduction are possible.

In Section II, we introduce some basic definitions used throughout the paper. We present related work on logic level power management in Section III. Section IV gives an overview of the problem of FSM decomposition. We describe our method in Section V. We first describe the architecture that allows for the disabling of the inputs/state registers and present an algorithm for the selection of the states for each of the sub-FSMs. In Section VI, we provide a set of experimental results and end with some conclusions and discussion of future work in section VII.

II. Basic Definitions

We use the standard definition of finite state machines:

Definition 1: A finite state machine is defined in the standard way as a tuple $M = (\Sigma, \Delta, Q, q_0, \delta, \lambda)$ where Σ is a finite set of input symbols, $\Delta \neq \emptyset$ is a finite set of output symbols, $Q \neq \emptyset$ is a finite set of states, $q_0 \in Q$ is the "reset" state, $\delta(q, a) : Q \times \Sigma \rightarrow Q$ is the transition function, and $\lambda(q, a) : Q \times \Sigma \rightarrow \Delta$ is the output function.

The specification of the finite state machine is equivalent to the specification of a state transition graph (STG), where each state corresponds to one node in the graph, and there exists an edge e_{ij} between two states q_i and q_j with label a/b iff $\delta(q_i, a) = q_j$ and $\lambda(q, a) = b$.

Under specific input line probabilities, it is possible to compute the stationary state and transition probabilities. The stationary state probability, $P(q_i)$, is the probability of the finite state machine being in state q_i in a given clock cycle. In this work, we will always use the

DAC 98, San Francisco, California

ıbsolute transition probabilities, indicated as $P(e_{ij})$. $P(e_{ij})$ repre-ıents the probability that, at any specific time, transition e_{ij} is active. $P(e_{ij})$ can be computed using:

$$P(e_{ij}) = P(q_i) \times P(a) \quad (1)$$

vhere $P(a)$ represents the probability of the primary inputs combina-ions that trigger the transition e_{ij}.

For each state q_i we can write an equation:

$$P(q_i) = \sum_{k=1}^{|Q|} P(e_{ki}) \quad (2)$$

Ve obtain $|Q|$ equations out of which any one equation can be derived rom the remaining $|Q| - 1$ equations. Since at any specific time the nachine is in one and only one state, we have a final equation:

$$\sum_{k=1}^{|Q|} P(q_i) = 1 \quad (3)$$

This linear set of $|Q|$ equations can be solved to obtain the different $^{>}(q_i)$'s.

This system of equations is known as the Chapman-Kolmogorov quations for a discrete-time discrete-transition Markov process, and, nder some general conditions, has a unique solution [11].

III. Related Work

So called power management techniques that shutdown blocks of ardware for periods of time in which they are not producing useful ata are effective methods to reduce the power consumption of a cir-uit. Shutdown can be accomplished by either turning off the power ıpply or by disabling the clock signal. A system-level approach is to lentify idle periods for entire modules and turn off the clock lines for ıese modules for the duration of the idle periods ([4], Chapter 10).

Power management techniques have also been proposed at the ›gic level. The main difference is that the shutdown of hardware ; decided on every clock cycle. The technique we propose in this aper falls under this category. In this section we describe previously roposed methods.

. *Precomputation*

One of the first logic-level shutdown methods is *precomputa-on* [1]. In this method a simple combinational circuit (the precom-ıtation logic) is added to the original circuit. Under certain input ›nditions, the precomputation logic disables the loading of all or a ıbset of the input registers. Under these input conditions, no power dissipated in the portions of the original circuit with only disabled ·gisters as inputs.

The choice of the number of inputs to use for the pre-computation ·gic is critical. The more inputs used the highest the probability the ·ecomputation logic will be active, thus disabling logic in the origi-ıl logic block. However, this also increases the size of the precom-ıtation logic, a circuitry overhead that is active all the time, thus ·fsetting the gains obtained by disabling A a larger fraction of the ne.

Once the number of inputs to the precomputation logic is fixed, e input selection is based on the probability that the outputs can be ›mputed without the knowledge of a specific input, i.e., the size of e observability don't-care set. Inputs with lowest probabilities are lected to be in the precomputation logic.

B. Gated-Clock Finite State Machines

The *gated-clock finite state machines* approach [2] 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 fact that this procedure is only applicable to Moore FSMs can be very limiting. Techniques to locally transform a Mealy FSM into a Moore FSM are given in [2] so that the opportunity for gating the clock is increased.

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 "super-state" and then transitions between states in this cluster are no more than self-loops in this "super-state". The decision as to what states make up the "super-block" basically gives the opportunity to maximize the number of self-loops.

C. Selectively-Clocked Systems

The implementation of a FSM in terms of sub-FSMs with the objective of achieving low power consumption by only enabling the clock signal of the active sub-FSM has been proposed recently [3]. However, the approach followed is completely different from the one we are proposing in this paper.

The work presented in [3] is concerned with the construction of controllers obtained during the process of behavioral synthesis. In this process, mutually exclusive sections of computation operations are detected. Each of these sections is then implemented as a separate FSM. The overall controller is thus made up of a set of interacting FSMs. Since by construction no two FSMs can be active simultaneously, only the clock signal for one of the FSMs needs to be working, therefore achieving a significant power reduction as compared to a monolithic implementation of the controller.

The technique we describe in the present paper is more flexible in the sense that it is not tied to the structure of some behavioral-level circuit description. However, the overhead in our approach may be higher. In any case, during the synthesis process we can take advantage of both techniques as the FSM decomposition approach can be applied to each individual FSM generated with the approach of [3].

D. Decomposition By Choice Of Disjoint Encodings

An approach that is very closely related to our work is based on the selection of encodings that are mutually orthogonal. This method can also be viewed as a state transition graph decomposition approach [5], although the resulting hardware does not follow strictly the standard structure for FSM decomposition, as described in the next section.

In this method, the STG is partitioned in two (or more) sets of states. All the states in one of the sets are assigned encodings with the first bit at 0, while the states in the other set are assigned encodings with that bit at 1. The combinational logic can now be broken into two separate blocks: one that is active when the first state bit is 0, and one that is active when the first state bit is 1. Since only one of these blocks can be active at any time, the other block can be disabled, leading to potentially significant power savings, if the state partition is chosen carefully.

The authors propose two methods to chose the state partition. The first one is based on the computation of the co-factors of the sequential circuit with respect to one of the state bits, and arbitrarily splits

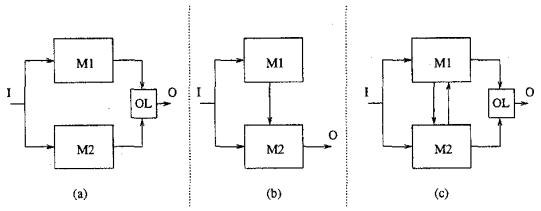

Fig. 1. Parallel, cascade and general approaches to finite state machine decomposition.

the set of two states into two disjoint sets. The second method is related to our own, in that it uses the stationary state probabilities to achieve a partition that leads to a final circuit with low power dissipation. However, the methodology used to implement the final circuit is very different, since only one set of registers is used, and therefore all the registers have to be clocked at every time instant.

IV. Decomposition Of Finite State Machines

The decomposition of finite state machines has been addressed by a number of authors, the first work dating back to 1960 by Hartmanis [7]. Several decomposition strategies have been proposed and are described in detail elsewhere [6]. Parallel and cascade decomposition, shown in Figure 1 (a) and (b), respectively, are simple, but of limited use with finite state machines that do not have a very regular structure.

On the other hand, the more general form of finite state machine decomposition, shown in Figure 1 (c), is applicable to any finite state machine. The structure of the resulting finite state machine is shown in Figure 2. Each of the sub-machines receives, as inputs, not only the

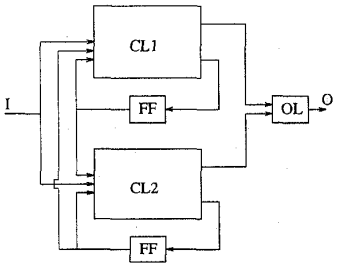

Fig. 2. Structure of decomposed finite state machine.

primary inputs and its own state variables, but also the state variables of the other sub-machine. Although there are several ways in which the STGs of each machine can be built, the following approach is simple and intuitive and will be used in this work.

1. Select a subset of the states in the original STG to belong to the first sub-machine, and let the remaining states belong to the second machine.
2. Generate two STGs, one for each sub-machine. 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 sub-machine, and one from the RESET state in the second sub-machine. The reverse is true for the symmetric case.

We illustrate the procedure with the following simple example de picted in Figure 3. Consider the following state transition graph for a simple sequence detector and assume that a decomposition of the cor responding FSM is desired, with states A and B to belong to machine M2, and all the other states to belong to machine M1.

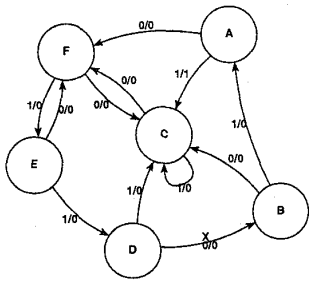

Fig. 3. State transition graph of the sequence detector.

The original STG is transformed into two smaller STGs. Tran sitions between the states in M1, as well as transitions between th states in M2, are copied without transformation. As shown in Fig ure 2, the number of inputs to each sub-FSM is enlarged with the valu of the state lines of the other sub-FSM, each FSM also providing a outputs the value of the state lines. Transition between a states in M

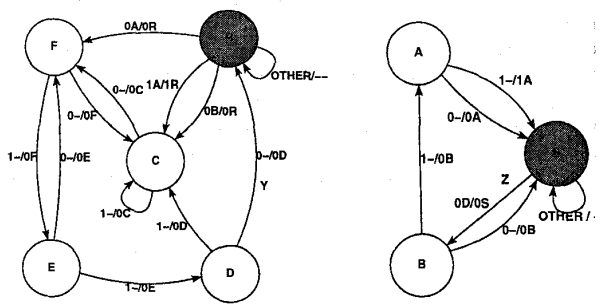

Fig. 4. STGs for the sequence detector after decomposition.

and a state in M2 require a slightly more complex treatment. Con sider, for example, the transition marked with **X**, in Figure 3, betwee states D and B. This transition corresponds to two new transition one in each of the STG of the sub-FSMs. In M1, it corresponds to th transition marked **Y**, between state D and state R (the newly create RESET state in M1). This transition is labeled with the original valu of the input (0) and has, as output, the value of the source state i M1, D. When machine M1 performs this state transition, it become inactive. In M2, it corresponds to the transition marked **Z**, betwee state S (again, the newly created RESET state in M2) and state E This transition is labeled with 0 D (the original value of the input plu the encoding of state D), and signals that machine M2 should becom active. All input combinations not explicitly shown in the edges lea ing the reset state of each machine are self-loops, and they signal th fact that the given machine is to stay inactive until it receives the rig input combination.

In practice, the procedure works in a slightly different way fro

he one illustrated here, because the codification of the sub-FSMs is ıot known at the time the STG is being partitioned. In fact, using the ;ame state codification for the sub-FSMS and for the original FSM vould be sub-optimal, specially in the case where one of the sub-'SMs is much smaller than the other one and requires a much smaller ıumber of registers. In the decomposition of large machines, this may epresent a significant advantage over an approach that uses the same et of registers to implement both sub-machines [5]. The exact way ıy which this information is exchanged between the sub-machines is letailed in section V-C.1.

V. Finite State Machine Decomposition For Low Power

The main objective of this work is to use finite state machine de-omposition techniques to achieve low power dissipation. The basic dea, described in this section, is to decompose the original machine nto two machines, one of them with a reduced number of states and ow power dissipation that performs needed computations for a large raction of the time.

A. Targeting The Partition Strategy For Low Power

Consider the decomposition shown in Figure 2 in Section IV and he state transition diagrams shown in Figures 3 and 4. Notice that, or any transition that takes place between two states other than the RESET state in the same factor machine, only this machine needs o be active. This means that, for these transitions, we can disable ntirely the other machine, avoiding all the power dissipation it incurs.

On the other hand, transitions that involve the RESET state and an-ther state are active simultaneously in both machines, because when ne machine is leaving the RESET state, the other one is entering it. 'hese transitions tend to generate the largest power dissipation, be-ause both machines are active at once.

The potential for the gains in power dissipation obtainable from ıe decomposition technique described in Section IV are larger if one elects 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.

The motivation for each of these factors is relatively straightfor-'ard. The first and second requirements specify that one of the ma-hines, the *small* machine, will be operating a large fraction of the me with a relatively small amount of power dissipation. The third :quirement minimizes the fraction of time both machines are operat-ıg, a situation that is undesirable because it corresponds to maximum ower dissipation.

Clearly, a partition that satisfies these requirements is not guaran-:ed to perform well, because the power dissipation of the final sys-:m depends on a variety of factors that cannot be considered at this ɔstraction level, like state encoding, combinational circuit synthe-s and technology mapping. However, the three requirements listed ɔove can be considered as useful heuristics in the selection of a par-tion that will achieve the desired reduction in power dissipation.

B. Selecting A State Partition For Low Power

The problem of selecting a graph partition that maximizes a given ɔst function has many applications and has been addressed in a vari-y of ways. One of the best known algorithms for this problem is the Kernighan-Lin algorithm [8]. This algorithm has been applied with success in a variety of applications, and was the algorithm used in this work.

The algorithm can be applied in the selection of both balanced or unbalanced partitions, with either a fixed or variable number of states in each partition. Assume, for the moment, that the number of states in each partition is fixed beforehand and that the STG has $N = |Q|$ states. We wish to obtain a partition in two sets with N_1 states on one set and N_2 states on the other set, with $N_1 <= N_2$.

The algorithm starts with an arbitrary partition that has two sets with, respectively, N_1 and N_2 states. It then selects a pair of states to swap, in a greedy way. Specifically, it selects the pair of states to be swapped that lead to higher increase (or smaller decrease) in the objective function. The states are then swapped, and the value of the resulting partition is recorded. These states are also recorded as already swapped, and are therefore fixed in their new partitions. After N_1 states have been swapped, the algorithm terminates, and outputs the best partition seen in the process.

B.1 Selection Of The Objective Function

The application of the Kernighan-Lin algorithm with a cost function that reflects the objectives stated above generates the desired finite state machine decomposition. The result can then be used to generate a sequential network that has a functionality equivalent to the original one, but dissipates, in general, less power.

Given the general objectives described above, we use the Kernighan-Lin algorithm to select a partition of the original set of states Q into two subsets Q_1 and Q_2, that maximize the following objective function:

$$F = \sum_{q_i \in Q_1, q_j \in Q_1} P(e_{ij}) - \beta\Big(\sum_{q_i \in Q_1, q_j \in Q_2} P(e_{ij}) + \sum_{q_j \in Q_1, q_i \in Q_2} P(e_{ij}) \Big) \quad (4)$$

The first summation represents the total probability of the transitions occurring entirely inside the STG of the *small* machine, while the second and third summations represent the total probability of the transitions occurring between the two machines.

Empirically, we found that values between 0.5 and 1.0 for the β coefficient worked best, and we selected a value of 0.7 for all experiments.

B.2 Complexity of the Kernighan-Lin procedure

The partition algorithm uses the standard data structures for the representation of state transition graphs. With this simple data structures, a total of $N_1 \times N_2$ pairings of states need to be considered for each step[1] of the Kernighan-Lin algorithm. The evaluation of the objective function can be done in time that is, under some general assumptions, linear on the number of states in the small partition, N_1. Finally, a total of N_1 steps need to be executed to finish the algorithm.

This means the procedure has complexity $O(N_1^3 N_2)$. However, N_1 is always a small number, because it is the number of states in the *small* machine. Typically, N_1 is never superior to 15, and N_1^3 is therefore bounded from above by a constant, leading to a total complexity that is linear in the total number of states in the original STG.

Empirically, we found that the procedure terminates in less than 30 seconds for all the circuits reported, taking less than a second for the majority of them. In any case, the execution of this procedure was never a bottleneck given that, in the majority of the circuits that we were unable to test, the STG extraction step was the limiting factor.

[1]Each step of this algorithm performs a state swap between the two partitions.

C. Generating The Final Network From The State Partition

The FSM decomposition method described in Section IV requires small changes to be effective in this application. These changes are motivated by the following facts:

- Because the *small* machine has a reduced number of states, it is possible to minimize the amount of information sent from the large machine to the small machine.
- Maximum power savings will be achieved if the machines are disabled when they are in a self-loop condition in the RESET state.

C.1 Minimizing Information Exchange Between The Two Machines

In the partition scheme described in Section IV and shown in Figure 1, the communication between the two sub-machines is completely symmetrical.

However, if one of the machines is much smaller than the other one (as measured by the number of states in the STG), this communication mechanism can be changed to our advantage in the following way:

- When the *small* machine makes a transition to the RESET state, the extra outputs of this machine encode information describing which state the machine was before reaching the RESET state.
- When the *large* machine makes a transition to the RESET state, the extra outputs of this machine encode information describing to which state the *small* machine should go from its RESET state.

Because the small machine has a smaller number of states, this change reduces the number of extra outputs and inputs required, thereby reducing the total power dissipation.

C.2 Generation Of The Disabling Network

In this application, it is important to have a sub-FSM disabled when it is in a loop in its reset state. Given the partition methodology outlined in Section IV, it is easy to generate an extra output for each sub-FSM that is used to enable the other machine.

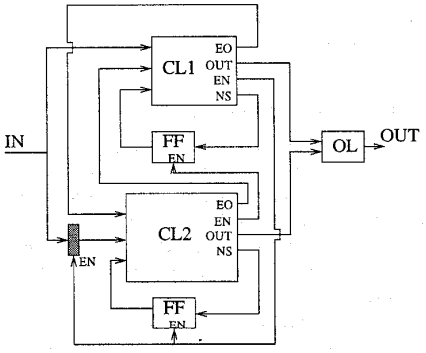

Fig. 5. Structure of decomposed finite state machine with enable circuits and blocking devices.

The structure of the decomposed machine with the enable circuits in place is the one shown in Figure 5. In this scheme, each machine generates a set of extra output variables (*EO*) that encode the information needed to transmit to the other machine which state it must go to, as described in Section V-C.1. Each sub-FSM also generates an enable signal (*EN*) that is used to disable the state registers in the other machine.

TABLE I
FINITE STATE MACHINE STATISTICS.

Circuit	# PI	# PO	# states	# literals	Power (μW)
opus	5	6	10	122	416
mark1	5	14	13	129	386
s386	7	7	13	190	422
sse	7	7	13	174	458
ex4	5	9	14	122	482
cse	7	7	16	255	407
kirkman	12	6	16	297	670
ex2	2	2	19	209	723
keyb	7	2	19	306	550
ex1	8	19	20	317	570
donfile	2	5	24	261	685
s820	18	19	25	433	808
s832	18	19	25	454	836
dk16	2	3	27	382	1087
styr	9	10	30	647	909
sand	11	9	32	718	1146
tbk	6	3	32	1019	1726
s510	19	7	47	381	831
planet	7	19	48	729	1832
s1488	8	19	48	942	1680
s1494	8	19	48	951	1675
scf	27	56	115	1058	1117

For one machine (M2, the *large* one), the primary inputs also g through a set of latches or AND gates, thereby stopping the propag tion of transitions from these inputs before they reach the combin tional logic. This is represented by the shaded block in Figure 5. W decided not to have the primary inputs of the *small* machine disable in the same way because this machine is, in general, small, and be cause the extra number of latches added to this small machine ma increase instead of decreasing the total power dissipation.

VI. EXPERIMENTAL RESULTS

We have applied our FSM decomposition technique on circuit from the MCNC91 and ISCAS89 benchmark set. We present a s of preliminary results on the power savings we have obtained. Th results were obtained with SIS running on a 170MHZ Ultra I Su Workstation, using the `power_estimate` command [13] to com pute the dissipated power. All circuits were first optimized usin `script.algebraic` and mapped using the `msu` library.

In Table I, we show statistics of the circuits we present resul for. The name, number of inputs, outputs and states for each of th circuits used is given in the first four columns. We give the number literals in column five, which is a good measure of circuit area. Th power dissipation of the original circuit is shown in the last colum assuming a supply voltage of 5V, a clock frequency of 20MHz, a zer delay model and uniform input probabilities.

Table II presents the results we obtained for each FSM. Since th algorithm for state selection described in Section V-B.1 is very eff cient, we run the algorithm for different values of n, the number states in the small sub-FSM. The cost function given in Eqn. 4 w used with $\beta = 0.7$, which we have found empirically to give best r sults. For each FSM in the benchmark set, we present results for th value of n with which the best power savings were obtained.

TABLE II
RESULTS OBTAINED AFTER DECOMPOSITION.

Circuit name	n	Blocking latches			Blocking ANDs		
		% A	Pwr	% P	% A	Pwr	% P
opus	4	47.4	277	33.5	42.2	281	32.6
mark1	6	55.3	274	29.1	49.1	304	21.4
s386	6	18.5	330	21.9	13.4	342	18.9
sse	5	40.3	231	49.6	34.5	252	44.9
ex4	5	30.5	268	44.2	24.0	234	51.5
cse	5	57.8	231	43.3	53.7	238	41.6
kirkman	3	17.6	461	31.3	10.3	420	37.3
ex2	8	52.6	437	39.6	51.0	462	36.0
keyb	8	27.2	343	37.7	23.7	271	50.7
ex1	6	38.1	459	19.6	34.7	467	18.2
donfile	5	54.8	485	29.2	53.5	574	16.2
s820	4	21.8	344	57.5	14.2	321	60.2
s832	4	19.6	442	47.1	12.3	429	48.7
styr	6	34.9	587	35.5	32.9	611	32.8
sand	15	42.5	1030	10.2	39.8	972	15.2
tbk	10	-22.1	928	46.2	-23.2	975	43.5
s510	7	45.7	372	55.3	37.8	224	73.1
planet	9	31.7	896	51.1	29.9	888	51.6
s1488	4	-1.1	345	79.5	-2.7	438	73.9
s1494	6	-3.4	392	76.6	-5.0	435	74.0
scf	14	47.1	582	47.9	42.8	734	34.3

We have experimented using latches and AND gates as blocking evices for the primary inputs (see Figure 5). In Table II, we give the ercentage area increase, power dissipation of the decomposed FSM nd the corresponding percentage power reduction using latches and nd gates. As we can observe from the table, impressive power sav- ıgs can be obtained, up to 80%. As it should be expected, the higher ıvings correspond to larger FSMs. In fact, for smaller FSMs there is o gain. This is due to the fact that the decomposed FSM has some xtra logic circuitry and two extra states, the RESET state in each sub- SM. This can represent a large overhead for small machines, but is ss important for larger FSMs.

As should be expected, using latches leads to larger increase in ea, with (sometimes significantly) higher power savings. This is not ue for all circuits as latches have larger input and internal capaci- ınces than AND gates, thus offsetting the reduced switching activity the outputs.

VII. CONCLUSIONS AND FUTURE WORK

We presented a methodology for the decomposition of finite state ıachines targeted towards low power dissipation. The methodology ses well known decomposition techniques to obtain a state machine at, for the majority of the large examples tested, exhibits a much naller power dissipation than the original. The results show that ower savings of up to 80% are possible in some of the examples.

Despite the good quality of the results obtained, there are several rections for future work that may improve these results and extend e range of applicability of the technique.

The single most important direction is probably the extension of is work to the case where the state transition graph cannot be ex- licitly described. In our experiments, we verified that the bottleneck sides in the extraction of the STG from the sequential circuit description. Because the decomposition algorithm works with an explicit description of the STG, it is not possible to apply the method to machines where the STG cannot even be extracted.

However, it may be possible to apply the decomposition idea even in this case, if a set of transitions with high enough probability exist to justify the method. The basic idea is that, in many cases, this set of transitions can be identified using Monte Carlo methods even without doing a complete traversal of the STG. With this set of transitions available, it is possible to perform a partition of the STG, considering only the states involved in these transitions and using only an implicit representation of the STG. Once the STG is partitioned, the rest of the method is directly applicable, making it feasible to use it in cases where the machine is too large to permit the extraction of the STG.

Another 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. It may be possible to modify the cost function described above to automatically include a term that depends on the number of states, thereby removing the need for the user to specify the value of this parameter or to perform a search for its right value.

Finally, it is clear that the results obtained in this paper can be improved if a more efficient implementation of the decomposition strategy is selected. In particular, it should be possible to reduce the overhead incurred by the addition of the extra outputs to each sub-FSM, by encoding these outputs in a different way that take into account the encoding of each of the sub-machines.

REFERENCES

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-Based Sequential Logic Optimization for Low Power. *IEEE Transactions on VLSI Systems*, 2(4):426–436, December 1994.

[2] L. Benini, P. Siegel, and G. De Micheli. Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines. *IEEE Transactions on Computer-Aided Design*, 15(6):630–643, June 1996.

[3] L. Benini, P. Vuillod, C. Coelho, and G. De Micheli. Synthesis of Low-Power Partially-Clocked Systems from High-Level Specifications. In *9th International Symposium on System Synthesis*, November 1996.

[4] A. Chandrakasan and R. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[5] S-H. Chow, Y-C. Ho, and T. Hwang. Low Power Realization of Finite State Machines – A Decomposition Approach. *ACM Transactions on Design Automation of Electronic Systems*, 1(3):315–340, July 1996.

[6] S. Devadas and A. Newton. Decomposition and Factorization of Sequential Finite State Machines. *IEEE Transactions on Computer-Aided Design*, 8(11):1206–1217, November 1989.

[7] J. Hartmanis. Symbolic Analysis of a Decomposition of Information Processing. *Information Control*, 3:154–178, June 1960.

[8] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, pages 291–307, February 1970.

[9] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. In *Proceedings of the International Conference on Computer-Aided Design*, pages 398–402, November 1993.

[10] F. Najm. A Survey of Power Estimation Techniques in VLSI Circuits (*Invited Paper*). *IEEE Transactions on VLSI Systems*, 2(4):446–455, December 1994.

[11] A. Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 2^{nd} edition, 1984.

[12] K. Roy and S. Prasad. Circuit Activity Based Logic Synthesis for Low Power Reliable Operations. *IEEE Transactions on VLSI Systems*, 1(4):503–513, December 1993.

[13] C-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. Despain, and B. Lin. Power Estimation Methods for Sequential Logic Circuits. *IEEE Transactions on VLSI Systems*, 3(3):404–416, September 1995.