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]). Re-timing 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 $\Sigma$ 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 $\lambda(q_i,a) = b$.

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

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

where $P(a)$ represents the probability of the primary inputs combinations that trigger the transition $e_{ij}$.

For each state $q_i$, we can write an equation:

$$P(q_i) = \sum_{k=1}^{Q} P(e_{ik})$$  \hspace{1cm} (2)

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

$$\sum_{k=1}^{Q} P(q_k) = 1$$  \hspace{1cm} (3)

This linear set of $|Q|$ equations can be solved to obtain the different $P(q_i)$’s.

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

III. RELATED WORK

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

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

. Precomputation

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. Under these input conditions, no power dissipated in the portions of the original circuit with only disabled gates as inputs.

The choice of the number of inputs to use for the precomputation logic is critical. The more inputs used the highest the probability the precomputation logic will be active, thus disabling logic in the original logic block. However, this also increases the size of the precomputation logic, a circuitry overhead that is active all the time, thus sacrificing the gains obtained by disabling $A$ a larger fraction of the time.

Once the number of inputs to the precomputation logic is fixed, an input selection is based on the probability that the outputs can be computed without the knowledge of a specific input, i.e., the size of the observability don’t-care set. Inputs with lowest probabilities are selected 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
We illustrate the procedure with the following simple example depicted in Figure 3. Consider the following state transition graph for a simple sequence detector and assume that a decomposition of the corresponding FSM is desired, with states A and B to belong to machine M2, and all the other states to belong to machine M1.

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 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.

The original STG is transformed into two smaller STGs. Transitions between the states in M1, as well as transitions between the states in M2, are copied without transformation. As shown in Figure 2, the number of inputs to each sub-FSM is enlarged with the value 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.

and a state in M2 require a slightly more complex treatment. Consider, for example, the transition marked X, in Figure 3, between states D and B. This transition corresponds to two new transitions, one in each of the STG of the sub-FSMs. In M1, it corresponds to the transition marked Y, between state D and state R (the newly create RESET state in M1). This transition is labeled with the original value 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 becomes inactive. In M2, it corresponds to the transition marked Z, between 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 plus the encoding of state D), and signals that machine M2 should become active. All input combinations not explicitly shown in the edges leaving the reset state of each machine are self-loops, and they signal that the given machine is to stay inactive until it receives the right input combination.

In practice, the procedure works in a slightly different way froi
he one illustrated here, because the codification of the sub-FSMs is not known at the time the STG is being partitioned. In fact, using the same state codification for the sub-FSMs and for the original FSM would 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 number of registers. In the decomposition of large machines, this may represent a significant advantage over an approach that uses the same set of registers to implement both sub-machines [5]. The exact way in which this information is exchanged between the sub-machines is detailed 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 decomposition techniques to achieve low power dissipation. The basic idea, described in this section, is to decompose the original machine into two machines, one of them with a reduced number of states and power dissipation that performs needed computations for a large fraction of the time.

1. Targeting The Partition Strategy For Low Power

Consider the decomposition shown in Figure 2 in Section IV and the 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 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 the 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 straightforward. The first and second requirements specify that one of the machines, the small machine, will be operating a large fraction of the time with a relatively small amount of power dissipation. The third requirement minimizes the fraction of time both machines are operating, a situation that is undesirable because it corresponds to maximum power dissipation.

Clearly, a partition that satisfies these requirements is not guaranteed to perform well, because the power dissipation of the final system depends on a variety of factors that cannot be considered at this abstraction level, like state encoding, combinational circuit synthesis and technology mapping. However, the three requirements listed above can be considered as useful heuristics in the selection of a partition that will achieve the desired reduction in power dissipation.

2. Selecting A State Partition For Low Power

The problem of selecting a graph partition that maximizes a given cost function has many applications and has been addressed in a variety 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 \( N_1 \leq N_2 \).

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:

\[
F = \sum_{q_1 \in Q_1, q_2 \in Q_2} P(e_{q_1}) - \beta \left( \sum_{q_1 \in Q_1} P(e_{q_1}) + \sum_{q_2 \in Q_2} P(e_{q_2}) \right)
\]  (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 \( \beta \) 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 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 \( O(N_1^2 N_2) \). 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 \( N_2^2 \) 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.](image)

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.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># PI</th>
<th># PO</th>
<th># states</th>
<th># literals</th>
<th>Power (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpus</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>122</td>
<td>416</td>
</tr>
<tr>
<td>markl</td>
<td>5</td>
<td>14</td>
<td>13</td>
<td>129</td>
<td>386</td>
</tr>
<tr>
<td>s386</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>190</td>
<td>422</td>
</tr>
<tr>
<td>sse</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>174</td>
<td>458</td>
</tr>
<tr>
<td>ex4</td>
<td>5</td>
<td>9</td>
<td>14</td>
<td>122</td>
<td>482</td>
</tr>
<tr>
<td>cse</td>
<td>7</td>
<td>7</td>
<td>16</td>
<td>255</td>
<td>407</td>
</tr>
<tr>
<td>kirkman</td>
<td>12</td>
<td>6</td>
<td>16</td>
<td>297</td>
<td>670</td>
</tr>
<tr>
<td>ex2</td>
<td>2</td>
<td>2</td>
<td>19</td>
<td>209</td>
<td>723</td>
</tr>
<tr>
<td>keyb</td>
<td>7</td>
<td>2</td>
<td>19</td>
<td>306</td>
<td>550</td>
</tr>
<tr>
<td>ex1</td>
<td>8</td>
<td>19</td>
<td>20</td>
<td>317</td>
<td>570</td>
</tr>
<tr>
<td>donfile</td>
<td>2</td>
<td>5</td>
<td>24</td>
<td>261</td>
<td>685</td>
</tr>
<tr>
<td>s320</td>
<td>18</td>
<td>19</td>
<td>25</td>
<td>433</td>
<td>808</td>
</tr>
<tr>
<td>s322</td>
<td>18</td>
<td>19</td>
<td>25</td>
<td>454</td>
<td>836</td>
</tr>
<tr>
<td>dk16</td>
<td>2</td>
<td>3</td>
<td>27</td>
<td>382</td>
<td>1087</td>
</tr>
<tr>
<td>styx</td>
<td>9</td>
<td>10</td>
<td>30</td>
<td>647</td>
<td>909</td>
</tr>
<tr>
<td>sand</td>
<td>11</td>
<td>9</td>
<td>32</td>
<td>718</td>
<td>1146</td>
</tr>
<tr>
<td>tbk</td>
<td>6</td>
<td>3</td>
<td>32</td>
<td>1019</td>
<td>1726</td>
</tr>
<tr>
<td>s510</td>
<td>19</td>
<td>7</td>
<td>47</td>
<td>381</td>
<td>831</td>
</tr>
<tr>
<td>planet</td>
<td>7</td>
<td>19</td>
<td>48</td>
<td>729</td>
<td>1832</td>
</tr>
<tr>
<td>s1488</td>
<td>8</td>
<td>19</td>
<td>48</td>
<td>942</td>
<td>1680</td>
</tr>
<tr>
<td>s1494</td>
<td>8</td>
<td>19</td>
<td>48</td>
<td>951</td>
<td>1675</td>
</tr>
<tr>
<td>scf</td>
<td>27</td>
<td>56</td>
<td>115</td>
<td>1058</td>
<td>1117</td>
</tr>
</tbody>
</table>

For one machine (M2, the large one), the primary inputs also go through a set of latches or AND gates, thereby stopping the propagation of transitions from these inputs before they reach the combinational logic. This is represented by the shaded block in Figure 5. We decided not to have the primary inputs of the small machine disable in the same way because this machine is, in general, small, and because the extra number of latches added to this small machine may 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 set of preliminary results on the power savings we have obtained. These results were obtained with SIS running on a 170MHZ Ultra I 55 workstation, using the powerset command [13] to compute the dissipated power. All circuits were first optimized using script_algebraic and mapped using the ncu library.

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

Table II presents results we obtained for each FSM. Since the algorithm for state selection described in Section V-B.1 is very efficient, we run the algorithm for different values of n, the number of states in the small sub-FSM. The cost function given in Eqn. 4 was used with β = 0.7, which we have found empirically to give best results. For each FSM in the benchmark set, we present results for the value of n with which the best power savings were obtained.
TABLE II
RESULTS OBTAINED AFTER DECOMPOSITION.

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>n</th>
<th>Blocking latches</th>
<th>% A</th>
<th>Pwr</th>
<th>% A</th>
<th>% P</th>
<th>Blocking ANDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>opum</td>
<td>4</td>
<td>47.4</td>
<td>277</td>
<td>33.5</td>
<td>42.2</td>
<td>281</td>
<td>32.6</td>
</tr>
<tr>
<td>marcl</td>
<td>6</td>
<td>55.3</td>
<td>274</td>
<td>29.1</td>
<td>49.1</td>
<td>304</td>
<td>21.4</td>
</tr>
<tr>
<td>s386</td>
<td>6</td>
<td>18.5</td>
<td>330</td>
<td>21.9</td>
<td>13.4</td>
<td>342</td>
<td>18.9</td>
</tr>
<tr>
<td>sse</td>
<td>5</td>
<td>40.3</td>
<td>231</td>
<td>49.6</td>
<td>34.5</td>
<td>252</td>
<td>44.9</td>
</tr>
<tr>
<td>ex4</td>
<td>5</td>
<td>30.5</td>
<td>268</td>
<td>44.2</td>
<td>24.0</td>
<td>234</td>
<td>51.5</td>
</tr>
<tr>
<td>case</td>
<td>5</td>
<td>57.8</td>
<td>231</td>
<td>43.3</td>
<td>53.7</td>
<td>238</td>
<td>41.6</td>
</tr>
<tr>
<td>kirkman</td>
<td>3</td>
<td>17.6</td>
<td>461</td>
<td>31.3</td>
<td>10.3</td>
<td>420</td>
<td>37.3</td>
</tr>
<tr>
<td>ex2</td>
<td>8</td>
<td>52.6</td>
<td>437</td>
<td>39.6</td>
<td>51.0</td>
<td>462</td>
<td>36.0</td>
</tr>
<tr>
<td>keyb</td>
<td>8</td>
<td>27.2</td>
<td>343</td>
<td>37.7</td>
<td>23.7</td>
<td>271</td>
<td>50.7</td>
</tr>
<tr>
<td>ex1</td>
<td>6</td>
<td>18.1</td>
<td>459</td>
<td>19.6</td>
<td>34.7</td>
<td>467</td>
<td>18.2</td>
</tr>
<tr>
<td>dontile</td>
<td>5</td>
<td>54.8</td>
<td>485</td>
<td>29.2</td>
<td>53.5</td>
<td>574</td>
<td>16.2</td>
</tr>
<tr>
<td>s802</td>
<td>4</td>
<td>21.8</td>
<td>344</td>
<td>57.5</td>
<td>14.2</td>
<td>321</td>
<td>60.2</td>
</tr>
<tr>
<td>s312</td>
<td>4</td>
<td>19.6</td>
<td>442</td>
<td>47.1</td>
<td>12.3</td>
<td>429</td>
<td>48.7</td>
</tr>
<tr>
<td>styr</td>
<td>6</td>
<td>34.9</td>
<td>587</td>
<td>35.5</td>
<td>32.9</td>
<td>611</td>
<td>32.8</td>
</tr>
<tr>
<td>sand</td>
<td>15</td>
<td>42.5</td>
<td>1030</td>
<td>10.2</td>
<td>39.8</td>
<td>972</td>
<td>15.2</td>
</tr>
<tr>
<td>tbr</td>
<td>10</td>
<td>-22.1</td>
<td>928</td>
<td>46.2</td>
<td>-23.2</td>
<td>975</td>
<td>43.5</td>
</tr>
<tr>
<td>s10</td>
<td>7</td>
<td>45.7</td>
<td>372</td>
<td>55.3</td>
<td>37.8</td>
<td>224</td>
<td>73.1</td>
</tr>
<tr>
<td>planet</td>
<td>9</td>
<td>31.7</td>
<td>896</td>
<td>51.1</td>
<td>29.9</td>
<td>888</td>
<td>51.6</td>
</tr>
<tr>
<td>s1488</td>
<td>9</td>
<td>-1.1</td>
<td>345</td>
<td>79.5</td>
<td>-2.7</td>
<td>438</td>
<td>73.9</td>
</tr>
<tr>
<td>s1494</td>
<td>6</td>
<td>-3.4</td>
<td>392</td>
<td>76.6</td>
<td>-5.0</td>
<td>435</td>
<td>74.0</td>
</tr>
<tr>
<td>scf</td>
<td>14</td>
<td>47.1</td>
<td>582</td>
<td>47.9</td>
<td>42.8</td>
<td>734</td>
<td>34.3</td>
</tr>
</tbody>
</table>

We have experimented using latches and AND gates as blocking
vices for the primary inputs (see Figure 5). In Table II, we give the
percentage area increase, power dissipation of the decomposed FSM
and the corresponding percentage power reduction using latches and
AND gates. As we can observe from the table, impressive power sav-
nings can be obtained, up to 80%. As it should be expected, the higher
vsings correspond to larger FSMs. In fact, for smaller FSMs there is
no gain. This is due to the fact that the decomposed FSM has some
strategic cility and two extra states, the RESET state in each sub-
SM. This can represent a large overhead for small machines, but is
not important for larger FSMs.

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

VII. CONCLUSIONS AND FUTURE WORK

We presented a methodology for the decomposition of finite state
machines targeted towards low power dissipation. The methodology
uses well known decomposition techniques to obtain a state machine
at, for the majority of the examples tested, exhibits a much
taller power dissipation than the original. The results show that
power 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
the 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-
citely described. In our experiments, we verified that the bottleneck
sides in the extraction of the STG from the sequential circuit de-
scription. 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 auto-
matic 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 im-
proved if a more efficient implementation of the decomposition strat-
agy is selected. In particular, it should be possible to reduce the over-
head 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

Precomputation-Based Sequential Logic Optimization for Low Power.

[2] L. Benini, P. Siegel, and G. De Micheli, Automatic Synthesis of
Low-Power Gated-Clock Finite State Machines. IEEE Transactions

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


State Machines - A Decomposition Approach. ACM Transactions on

[6] S. Devadas and A. Newton, Decomposition and Factorization of Se-
quential Finite State Machines. IEEE Transactions on Computer-Aided


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

for Low Power. In Proceedings of the International Conference on

December 1994.


[12] K. Roy and S. Prasad, Circuit Activity Based Logic Synthesis for
Low Power Reliable Operations. IEEE Transactions on VLSI Systems,

Power Estimation Methods for Sequential Logic Circuits. IEEE Trans-

763