Low power finite state machine synthesis using power-gating

Sambhu Nath Pradhan\textsuperscript{a,b}, M. Tilak Kumar\textsuperscript{b}, Santanu Chattopadhyay\textsuperscript{b}

\textsuperscript{a} Department of E and ECE, IIT Kharagpur 721302, West Bengal, India
\textsuperscript{b} Department of ECE, NIT Agartala, 799055 Tripura, India

A R T I C L E   I N F O

Article history:
Received 2 March 2010
Received in revised form 26 November 2010
Accepted 3 March 2011
Available online 21 March 2011

Keywords:
Low power
Finite state machine
Power-gating
Genetic algorithm

A B S T R A C T

Power-gating turns off the power supply of a portion of the circuit completely, resulting in total elimination of power consumption for that part. However, it also necessitates that the sub-circuit to be activated should be charged for some time before its activation. This critical issue can influence the decomposition of a finite state machine (FSM) for its power gated implementation. In this paper we have presented a power-gating method that integrates FSM partitioning with state encoding, thus providing a total solution to the problem of power-aware FSM synthesis. It shows better results, in terms of dynamic and leakage power consumption, compared to the existing techniques reported in the literature.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Many of the emerging computing and communication equipments are control dominated. Even for designs containing a good number of datapath elements, a sizeable portion is occupied by the controller. As the devices are mostly portable and hand-held, reducing power dissipation (and associated heat generation and cooling arrangement overheads) has emerged as the primary concern of today’s VLSI designers. While the datapath elements can be shutdown when they are not being used, controllers are always active. As a result, a good amount of system power is consumed by the controller. Controllers are mostly implemented as finite state machines (FSMs). Thus, power-efficient synthesis of FSM has come up as a very important problem domain, attracting lot of researchers to work on it. This paper is an attempt to develop an approach to synthesize FSMs consuming less power. It uses power-gating technique that can reduce both dynamic and leakage power consumed by the circuit.

State assignment is one of the most crucial steps towards the synthesis of FSMs. As a result, a lot of work has been done in FSM synthesis and state encoding targeting low power [1–5]. A physical partitioning of FSM has been proposed in [1] with considerable increase in area of the circuit. The traded-off area results in dynamic power saving via switching activity reduction. However, leakage power, being dependent on the area of the circuit, may increase significantly. The work [2] presents a state encoding algorithm that minimizes the Boolean distance between the codes of states with a high transition probability, using a probability description of the FSM. This work also lacks leakage power handling. The works [6,7] discuss about low power FSM synthesis using an appropriate mix of D- and T-flip-flops. As a judicious mix is used, combinational area and power get reduced. However, T-flip-flops require more area than D. Thus, area may increase, affecting the leakage power consumption. In [8,9], partition directed state-encoding algorithms have been presented. Both the approaches target only dynamic power saving. Moreover, in [8], the area overhead is about 77% of the normal FSM implementation. State decomposition has been addressed in [10]. The work is not generic in nature as it only looks into the FPGA based realization of the resulting circuit. The work presented in [11] exploits a fundamental and important source of power saving—shutting down useless parts of a sequential circuit. This can lead to both dynamic and leakage power reduction. But the work does not provide any detailed experimental results on power savings. In [12,13], clock gating technique has been reported for low power FSM decomposition. Clock-gating can reduce dynamic power considerably, but leakage power cannot be controlled via this technique. All these works ignored the leakage power consumed by the circuit under the premises that leakage power is quite insignificant as compared to the dynamic power. On the other hand, the International Technology Roadmap for Semiconductors (ITRS) projects an exponential increase in the leakage power with miniaturization of devices [14]. The present paper addresses both dynamic and leakage power saving using power-gating.

Power-gating is a technique for saving both leakage and switching power by shutting off the idle blocks of the circuit. Now-a-days integrated power gating has emerged as a primary knob for balancing the needs for high performance and low standby power during periods of circuit inactivity. Many works have incorporated...
power-gating as the tool to reduce power. An analysis of the efficiency of power-gating for Clocked Storage Elements (CSEs) has been presented in [15]. It examines the energy savings for standard clock gated CSEs versus their power gated counterparts. The application of power gating circuits to semicustom design based on standard-cell elements is given in [16]. Authors propose a new power network architecture that enables the use of conventional standard-cell elements for the combinational logic. Power gating for the structured ASIC, implemented using existing standard cell design tools, has been introduced in [17]. Designing a power-gating structure with high performance in the active mode and low leakage and short wakeup time during standby mode is an important and challenging task. The works [18,19] address this issue. Authors in [19] present a tri-modal switch cell that enables the implementation of multimodal power gating, including active, data-retentive drowsy, and deep sleep modes. A per-core power gating architecture for multicore processors is presented in [19]. Impact of power gating depends on the workload running on the processor. Power saving employing power gating with varying work-load on processors has been explored in [18,20]. Another variation in power gating design, namely, reconfigurable power gating, has been utilized in the design of precision multiplier in [21]. ITRS roadmap has projected that in the submicron era, the delay of an interconnect wire segment is expected to dominate over the delay of logic blocks. Power gating for the wire in the form of bus segmentation has been explained in [22]. Power-gating structure for the reduction of peak current and voltage glitch in System-on-Chip environment has been presented in [23]. Power gating with multiple sleep modes has been proposed in [24].

All the power gating techniques discussed above consider micro-architectural level of circuit and system design. There is not much work reported on the application of power gating in the higher level of system design in the form of finite state machine (FSM) synthesis. The work [25] proposes a power-gating technique for FSM decomposition targeting low power. The FSM is partitioned into two or more sub-machines, only one of which is active most of the time. Power supply of other sub-machines can be cut off to save energy. Since adjustment of supply voltage may not finish instantly, for activating a sub-machine, it is needed to raise the supply voltage ahead of time. This makes the partitioning problem complicated. In [25], a simulated annealing based algorithm has been proposed to perform the decomposition without timing penalty. However, the method presented in [25] does not consider state encoding. Detailed experimental results are also not presented. In our paper, we consider the problems of FSM decomposition and state encoding together, targeting low power dissipation in power-gating technique. In the light of the above discussion, the salient contributions of this paper are as follows:

1. It integrates the two problems, partitioning and state encoding, into a genetic algorithm based formulation.
2. For the resulting partitions and encoding, it compares the following:
   (a) Dynamic power consumed by the combinational part of the circuit as estimated by SIS [26] with the same resulting from NOVA [27].
   (b) Total power for the partitions as obtained via synthesis using Synopsys Design Vision tool, with the same for circuits resulting from state encoding techniques like NOVA [27], JEDI [28]. This includes both dynamic and leakage power.
   (c) Assuming the combinational logic to be realized as 2-level Pseudo-NMOS PLA, the leakage and switching activity has been computed and compared with NOVA encoded circuit. The same can be done with other design styles as well, such as, static and dynamic CMOS, once the corresponding library characterizations are performed.
   (d) Power has also been compared with other existing low-power state encoding approaches.

The rest of the paper is organized as follows. In Section 2 we have presented the power-gating structure. Section 3 illustrates the FSM partitioning and encoding strategy using a genetic algorithm (GA) approach. In Section 4, the modeling strategy for dynamic and leakage power has been given. Section 5 details the experimental results of our power-gating technique and compares with the existing works. Finally, Section 6 notes the conclusion.

2. Power-gating implementation

A common way to shutdown the supply voltage for part of a circuit is to use sleep transistors (Fig. 1). A sleep transistor is added between Vdd and the circuit (or between the circuit and ground). The drain electrode of the sleep transistor forms a virtual Vdd, the voltage of which is grid-controlled.

An FSM is defined as a tuple $M = (S, \Sigma, A, \delta, Q, S_0)$ where $S$ is the finite set of states, $\Sigma$ is the finite set of inputs, $A$ is the finite set of outputs, $\delta: S \times \Sigma \rightarrow S$ is the state transition function, $Q: S \times \Sigma \rightarrow A$ is the output function and $S_0$ is the initial state. A finite state machine can be represented using state transition graph (STG), denoted as $G = (V, E)$, where $V$ is the set of the vertices and $E = \{(a,b) | a, b \in V \}$ is the set of the edges. Each node in $V$ corresponds to a unique state and the edges represent transitions between the states.

Since FSMS are used mostly to realize the controller portion of a system, it is very much essential to have good power-aware design of them. The FSM may consist of a large number of states—the states often forming clusters. The controller normally operates within a cluster, switching to a new cluster at some point of time. Thus, a power-aware FSM synthesis strategy can exploit this situation and realize the clusters separately. This enables shutting down the power to the cluster(s) which is (are) not active. However, in case of inter-cluster transitions, special care needs to be taken to avoid the delay in waking up the circuit. In this work, we have considered a bipartitioning strategy to achieve power minimization via power-gating.

2.1. Partitioning strategy

The scheme has been shown in Fig. 2. Here, a given FSM $F$ with $n$-states $S = \{s_1, s_2, ..., s_n\}$ is partitioned into two subFSMs $F_1$ (with states $S_1 = \{s_1, s_2, ..., s_{l}\}$) and $F_2$ (with states $S_2 = \{s_{l+1}, s_{l+2}, ..., s_n\}$) such that $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$. As shown in Fig. 2, for both the FSMs, there is a common state register holding the state bits. If set $S_1$ has $k$-states and $S_2$ has $l$-states, then the number of state-bits needed to represent the states is given by, $(1 + \max\{\log k, \log l\})$. The extra bit is used to identify the machine whose output determines the primary outputs and the next-state values. The logic COMB1 realizes the combinational part of $F_1$ while that of $F_2$ is realized by COMB2. Power-gating transistors are inserted in the supply lines of these blocks. As we will see later, it is sometimes

![Fig. 1. PMOS and NMOS based power-gating design.](image-url)
necessary that both the machines be ON simultaneously for some states of the FSM, while for the remaining states, only one of the two submachines needs to be ON. This has been accomplished by the "Enable logic" block. Depending upon the values present in the present state lines, it asserts EP₁ and/or EP₂ allowing the \( V_{dd} \) supply to reach the corresponding combinational logic. For the submachine active at a time instant, the latch at its input is enabled (via signal LE) to allow the primary input changes and state transitions reach the corresponding combinational logic. It may be noted that even when both the submachines are powered, it is only one of these that computes the primary output and the next-state values. Multiplexers are used at the outputs of the combinational blocks to select the proper primary outputs and next state bits.

Next, we state some of the definitions [25] to enable us to formulate the partitioning and state assignment problem.

- **Inner transition:** a transition from one state to another, where both the states belong to the same submachine.
- **Cross transition:** a transition from one state to another, where the states belong to different submachines.
- **c-Expansion:** a c-expansion of a state is the set of the states that can be reached from it within c steps.
- **Boundary depth:** it is defined as the number of clock cycles needed to turn ON the submachine to be activated.
- **Boundary states:** a boundary state between two submachines is a state in one submachine, which is within the boundary depth of another machine. We use \( D(F₁, F₂) \) to denote the set of boundary states in \( F₁ \) leading to \( F₂ \). The sum of the boundary state probabilities of \( D(F₁, F₂) \) is denoted as \( P_{bd}(F₁, F₂) \). Similarly, the sum of the boundary state probabilities of \( D(F₂, F₁) \) is denoted as \( P_{bd}(F₂, F₁) \).

**Example.** Fig. 3 is an example showing the bipartitioning of a state transition graph (STG) into \( F₁ \) and \( F₂ \). For this example, the transitions between the states of \( F₁ = \{S₁, S₂, S₃, S₄, S₅, S₆, S₇, S₈, S₉, S₁₀, S₁₁, S₁₂\} \) and the transitions between the states of the submachine \( F₂ = \{S₃, S₄, S₅, S₁₀, S₁₁\} \) are the inner transitions. The transitions from state \( S₃ \) to \( S₄ \), \( S₂ \) to \( S₃ \), \( S₁ \) to \( S₂ \), \( S₁₀ \) to \( S₁₁ \) to \( S₁₂ \) to \( S₁₃ \) to \( S₁₄ \) are the cross transitions. If the boundary depth is taken as 2, then the set of boundary states in the subFSM \( F₁ \) is \( D(F₁, F₂) = \{S₃, S₄, S₅, S₁₂\} \) and for \( F₂ \) \( D(F₂, F₁) = \{S₃, S₄, S₅\} \). When the circuit is in a boundary state, both of the submachines need to be turned ON because of the recovery time needed to turn ON the machine. In such cases, excessive energy will be consumed. Therefore, it is important to reduce the probability that the FSM is in a boundary state. In order to evaluate the quality of a bipartition, we estimate the power consumption of the overall circuit. The total power can be stated as

\[
\text{Gated power} = P(F₁) \times \text{Power}(F₁) + P(F₂) \times \text{Power}(F₂) + P_{bd}(F₁, F₂) \times \text{Power}(F₁) + P_{bd}(F₂, F₁) \times \text{Power}(F₁)
\]
where, \( P(F_1) \) is the probability that the current state belongs to \( F_1 \), \( P(F_2) \) is the probability of the present state being in \( F_2 \), \( P(F_1,F_2) \) and \( P(F_2,F_1) \) are as defined earlier. The first two terms of Eq. (1) are the contributions of submachines that are active. The last two terms represent the power consumed when a submachine is not active but is in the recovery process.

The quality of the power-gating circuit depends on the partitioning and encoding of the states of FSM. We have used a genetic algorithm [29] based strategy for partitioning of FSM states and their encoding. The following section describes this partitioning and state encoding approach.

### 3. Genetic algorithm for partitioning

Genetic algorithms (GAs) [29] are a class of related stochastic search and optimization techniques. GAs are capable of operating within domains that are traditionally thought to be difficult to optimize. It is a global optimization technique that is able to identify globally optimal or near optimal solutions. The key idea behind GAs is to emulate the way nature uses evolution. By mimicking natural genetic processes, GAs are able to evolve solutions to real world problems, if they have been suitably encoded. GAs work with a population of individuals, each of which represents a possible solution to a given problem. A population of a fixed number of chromosomes is maintained. Each chromosome is assigned a fitness score according to the merit of the corresponding solution. Individuals with a high fitness score are given opportunities to reproduce with other members of the population. This produces new individuals as ‘offspring’, who display features that are taken from each parent. The least fit members of the population are less likely to be selected for reproduction, and so they will die out. By favouring mating between the more fit individuals, the most promising areas of the search space are explored. If the GA has been designed correctly, the population will converge to an optimal solution to the problem. The problem solution using a evolutionary GA is as shown in Fig. 4. A standard GA proceeds as follows:

1. Randomly generate an initial population.
2. Define the selection criteria of the chromosomes for reproduction.
3. Determine the fitness value of each chromosome in the current population.
4. Produce the offspring in the presence of variation inducing operators such as mutation and crossover.
5. Repeat steps 2, 3 and 4 until a satisfactory solution is obtained.

In order to apply a GA to any optimization problem, it is necessary to define an individual solution representation (that is, chromosome), mutation operator, crossover operator, fitness function and selection procedure. Next, we enumerate the same for the partitioning and state assignment problem.

Solution structure: we have used a chromosome with two parts—the partition part and the state encoding part. For an \( N \)-state FSM, the partition part is an \( N \)-bit array, whereas, the state encoding part is an array of \( N \) integers. Corresponding to the \( i \)th state, the partition bit identifies the partition to which the state belongs (partition 1, if the bit is ‘0’ and partition 2, otherwise). The state encoding part identifies the codes assigned to the states. The most significant bit (MSB) of state code is set to ‘0’ for states in partition 1, whereas, the bit is set to ‘1’ for states in partition 2.

<table>
<thead>
<tr>
<th>Genetic operators: three genetic operators, namely, crossover, mutation and direct copy have been used to evolve the new generations.</th>
</tr>
</thead>
</table>

Crossover: to select chromosomes participating in crossover, first, the whole population is sorted according to the fitness value. A certain percentage of population (here, 20%) with better fitness value is considered to be the “best class”. Population size is taken to be 100. To select a chromosome participating in crossover, first a random number between 0 and 9 is generated. If the number is less than 6, a chromosome from the best class is selected randomly. Otherwise a chromosome is selected from the entire population. This approach of selecting more fit chromosomes to participate in crossover leads to the generation of better off-springs as compared to the truly random ones. Similarly, the other parent gets selected. Next, a parametric crossover [29] is used to create an off-spring. A random number between 0 and 9 is generated. If the number is above 6, the new chromosome is created with first position same as the first parent, otherwise, it is taken from the second parent’s first position. Similarly, all the positions of the newly created chromosome are filled by generating random numbers. The resultant chromosome is modified to satisfy the requirement that each entry in the state code part be
unique. 70% of the total population has been generated via the crossover operation.

**Mutation:** to perform the mutation, two random numbers are generated between 1 and the width of the chromosome. Between these two generated positions, the entries are toggled. The positions are changed from 0 to 1 or 1 to 0 in the first part while the second part positions are divided by 2. The resulting chromosome is modified to satisfy the uniqueness criteria. Here, we have taken mutation rate as 10% of the total population.

**Direct copy:** the basic GA has been modified so that the solution does not degrade between the generations. To ensure this, 20% best chromosomes are directly copied to the next generation.

**Termination:** the GA terminates when there is no improvement in result over the previous 40 generations.

**Fitness measure:** as we are doing power optimization, the fitness should reflect the power consumption. For this purpose, the combinational logic part corresponding to the two partitions are extracted, minimized using the two-level minimizer Espresso [30], and the corresponding dynamic power values are computed using the “power-estimate” command of SIS [26] with appropriate options. These two values are then combined together using Eq. (1). It may be noted that though total power of a circuit has contributions from both dynamic and leakage components, we have used the estimate of dynamic power only. This simplifies the fitness calculation. Moreover, as the idle component is getting switched off, the leakage power will also get reduced considerably. This has been readily reflected in our experimental results.

## 4. Power modeling

In this section, we present our model to get the estimate of dynamic and leakage power consumption in the final resulting circuit for the sake of comparison with other FSM synthesis techniques. We assume that the combinational logic has been implemented in a two-level Pseudo-NMOS style. It may be noted that our fitness calculation targets two-level realization, and that PLA’s are mostly implemented in Pseudo-NMOS NOR-NOR style. However, other design styles can be followed as well with proper power modeling.

### 4.1. Switching power estimation

#### 4.1.1. Dynamic power of combinational part

To estimate the dynamic power of the combinational logic, we need to compute the expected switching activity of the logic gates. We assume that primary inputs are uncorrelated and are statically independent of each other. In this case, the probability of a primary input being ON is 0.5. To compute the ON-probabilities of present state lines, first we compute the steady-state probabilities of individual states of FSM. For this, the FSM has been modeled as a Markov chain [31], a set of linear Chapman–Kolmogorov equations are formulated with normalization criteria, and are solved. This is given in the following.

**4.1.1.1. Steady state probability calculation.** It is possible to compute the steady state probabilities of the states and the transition probabilities from the specific input line probabilities. The steady state probability $P(s_j)$ is the probability of the finite state machine being in the state $s_j$ at a time instant. The state transition probability for the transition from $s_i$ to $s_j$ is defined as $P(s_j | s_i)$ and is computed as follows:

$$P(s_j | s_i) = P(s_j) \times P(k)$$

where $P(k)$ represents the probability of the primary input combination holding true for which the transition from $s_i$ to $s_j$ takes place. The steady state probabilities of the FSM states are calculated using the following equations:

$$\sum P(s_j | s_i) = 1 = \sum P(s_j)$$

This system of equations is known as Chapman–Kolmogorov equations [32] for a discrete-time discrete-transition Markov process. By solving this set of linear equations using the Gauss–Jordon elimination method, the steady state probabilities have been determined.

Now, the probability of a particular state-bit line being 1 is taken to be equal to the sum of steady-state probabilities of all those states whose codes have this particular bit turned ON. Moreover, while computing the power of the combinational logic of the first machine, the states belonging to it and those belonging to the boundary states of second machine are considered. It may be noted that the machine is ON in only these states. Similarly, for the combinational logic of second machine, the states of it along with the boundary states of first machine are considered. A similar procedure has been used to compute the OFF probabilities of state-bits feeding individual combinational logic. It may be noted that the sum of the ON and OFF probabilities of the present state lines are not equal to 1, but they are equal to machine ON probability. All the present state line probabilities are then normalized with the machine ON probability to make the sum of the ON and OFF probabilities of the state lines equal to 1.

The ON-probability of a NOR gate in the first level of combinational logic can be computed from the OFF probabilities of all inputs feeding the gate. For this, OFF probabilities of the primary inputs are taken as 0.5, while for the state-bit lines, it is computed by the procedure noted above. Once the ON-probabilities of each of the NOR gates in the first level are known, the switching activities can be computed. Summing up the switching of all such gates, we get the total switching activity of the first level NOR gates. For the second level NOR gates, the ON-probabilities are computed by finding the ON-probabilities of first level NOR-gates feeding them. Once the probabilities are known, switching activities can be obtained. Summing up the switching activities of first and second level gives the total switching activity.

#### 4.1.2. Dynamic power of sequential part

To find the dynamic power of the sequential part of individual machines, all unreachable states are first eliminated, and the graph is transformed into a weighted undirected graph $G$ by collapsing all multiple-directed edges between states $s_i$ and $s_j$ into a single undirected edge whose weight $w_{ij}$ is the sum of the directed total transition $(w_{ij})$ probabilities between these two states. The dynamic power consumption is then given by [33]

$$C_{dynamic} = \sum_{ij} w_{ij} H(C_i, C_j)$$

where $C_i$ and $C_j$ are the binary codes assigned to states $s_i$ and $s_j$, respectively, and $H(C_i, C_j)$ is the Hamming distance between $C_i$ and $C_j$.

### 4.2. Leakage power estimation

#### 4.2.1. Leakage power of combinational part

To calculate leakage of a NOR-gate, we have used the runtime mode leakage considering all input probabilities. Simulation for leakage power has been carried out using CADENCE SPECTER at 90 nm UMC technology. All the transistors are of minimum size supported at 90 nm technology. The transistors are special purpose transistors. These are low power, low leakage transistors available in UMC 90 nm Technology library. Probability
dependent leakage power is given by

\[ P_{\text{leakage}} = V_{dd} \sum S_k \times I_k \]  

(4)

where, \( k \) ranges over all possible input states of the NOR gate, \( S_k \) is the probability of state \( k \) and \( I_k \) is the leakage current of state \( k \). \( V_{dd} \) is the supply voltage.

If the number of inputs to a NOR gate is very large, direct application of Eq. (4) to estimate leakage power becomes infeasible. So we have estimated leakage in a different way as follows. Simulation of up to 30 input NOR gate has been carried out. Length and width of pull up transistor are 500 and 120 nm, respectively. All the pull down transistors has length and width of 90 and 135 nm, respectively. In all the cases, it has been observed that when all the inputs are zero, leakage power is approximately equal to the number of inputs times 4.518 pW, which is the off state leakage power of single input NOR gate. It is also observed that the leakage power of all non-zero patterns is very close to the leakage power of any NOR gate is thus given by,

\[ \text{leakage power due to all zero inputs} \]

\[ + \text{(1-all zero probability) \times average leakage value} \]

4.2.2. Leakage power of sequential part

Consider a general sequential circuit with \( l \) inputs, \( O \) outputs, \( V=\{v_1, v_2 \ldots, v_m\} \) present-state variables, and \( U=\{u_1, u_2 \ldots, u_n\} \) next-state variables. A transition from state \( s_a \) to state \( s_b \) implies the leakage power [33] in the sequential part of the FSM as

\[ \text{Leakage}(s_a, s_b) = \sum (X. \text{LeakTable}(clk = 1, output = v_i, input = u_i) + (1-X). \text{LeakTable}(clk = 0, output = v_i, input = u_i)) \]

where, \( X \) is the fractional duty cycle, \( v_i \) is bit \( i \) of \( s_a \), and \( u_i \) is bit \( i \) of \( s_b \). \text{LeakTable} provides values for static power consumption in a given flip-flop as a function of inputs and state. A static cost metric for the FSM (sequential part) can be formed as the sum of all leakage values between two states \( s_a \) and \( s_b \) each weighted with the probability of being at state \( s_a \) and having \( s_b \) as the next state. This probability, represented as \( P_{ab} \), can be computed as the steady state probability of state \( s_a \) multiplied by the transition probability from state \( s_a \) to state \( s_b \). The leakage power of the sequential part is calculated as follows:

\[ C_{\text{static}} = \sum \sum (P_{ab} \times \text{Leakage}(s_a, s_b)) \]

Since we are targeting low power realization, instead of ordinary flip-flop, we have used the Hybrid Latch Flip Flop (HLFF), shown in Fig. 5 [34]. The structure has been simulated using the CADENCE 90 nm technology and the corresponding leakage values are shown in Table 1.

5. Experimental results

The proposed power-gating decomposition technique has been implemented in C language on a Pentium 4 machine with 3 GHz clock frequency and 1 GB main memory. Table 2 presents the combinational power results obtained by our approach. The details of the columns of Table 2 are as follows. Column \( I/O/S \) notes the number of primary inputs, outputs and states. Column \( P_{FL}(F_1, F_2) \) is the sum of the steady state probabilities of all the boundary states in \( F_1 \). Similarly, \( P_{FL}(F_2, F_1) \) represents the sum of the steady state probabilities of all the boundary states in \( F_2 \). Column \( P(F_1) \) represents the steady state probability of the states in \( F_1 \). The power-gating results are obtained by estimating the power consumed by the combinational logic of \( F_1 \) and \( F_2 \), respectively, and summing them using Eq. (1). For the power calculation we have used the ‘power_estimate’ command of SIS [26]. SIS assumes a supply voltage of 5 V and operating frequency of 20 MHz. It may be noted that SIS gives only the dynamic power estimation (it does not provide the leakage power). Hence, Table 2 notes the dynamic power only for 2-level realization of the resulting combinational circuits. Circuit NOVA notes the power required by the combinational logic generated by the state assignment tool NOVA [27] with ‘–e ioh’ option. The result shows on an average 35% saving in dynamic power. The maximum savings are for circuits like s820 (78%), s832 (76.9%), s510 (58.6%). Saving is higher for the cases with lower boundary state probabilities. For FSMs with higher boundary state probabilities, both the machines are always on, thus requiring higher power. Circuits, such as s820, s832 and s510 have lower boundary state probabilities, so lower power dissipation. The last column gives the number of generations used in GA beyond which no further improvement in the fitness function could be observed. To see the impact of power-gating, we have formulated another GA in which only state-encoding has been performed for the whole FSM—it does not do any partitioning. The dynamic power consumed by the combinational part of the resulting circuit has been estimated using SIS and noted in the column marked GA of Table 2. As it can be computed from the table, the GA along with power-gating requires 25.5% lesser dynamic power than the GA that performs state encoding only. This clearly establishes the impact of power-gating. The column \( \text{COMB delay} \) notes the delay of the combinational block corresponding to the state assignment given by NOVA. The column \( \text{GATED delay} \) notes the maximum of the delays

![Fig. 5. Schematic of Hybrid Latch Flip Flop (HLFF).](image-url)
of two combinational blocks, COMB1 and COMB2, corresponding to the two partitions generated by our tool. In 13 out of 16 circuits, the partitioned realization has resulted in delay reduction.

Next, we have carried out another set of experiments taking the final encoded submachines of each FSM. Before synthesis, both the submachines and original machine are converted into Verilog HDL codes. The two submachines of each FSM are synthesized separately in Synopsys Design Vision [35] to get the dynamic and leakage power of the synthesized circuit. Using Eq. (1), dynamic and leakage power of the total FSM are calculated. This dynamic power is given in the column GATED_SW of Table 3. The leakage power is given in the column GATED_LEAK. The technology in Synopsys Design Vision synthesis tool is 90 nm Faraday, where the global operating voltage is 1 V. It may be noted that the tool synthesizes the circuit into a multilevel one.

The original FSM is then encoded with the state assignment tool NOVA [27]. This NOVA encoded circuit is also synthesized in Synopsys Design Vision tool at 90 nm Faraday technology. Switching and leakage power after synthesis in Synopsys tool for the NOVA encoded circuit are given in columns NOVA_SW and NOVA_LEAK of Table 3, respectively. For the circuits s208, sand, s386 and s510 switching power savings are 77.41%, 67.84%, 59.86% and 64.38%, respectively. Some circuits like, dk27, ex4, s1 and dk512 do not exhibit switching power saving. For dk27 leakage dissipation is also high. This happens because the GA produced solutions are optimized for two-level realization. However, since a good two-level minimization can act as the basis of multilevel minimization also, as shown in Table 3, we can get 15.3% average switching power saving. The circuits, which show maximum switching power saving also give maximum leakage power saving. Leakage power saving for the circuits s208, sand, s386 and s510 are 67.5%, 56%, 52% and 48%, respectively. Average leakage power savings in circuits synthesized using our approach is about 9% compared to the circuits encoded using NOVA. The fourth and the last columns of Table 3 present the power of the encoded FSM using JEDI [28] assignment tool that targets multi-level realization. It shows power improvement in some of the circuits like s208 (79.87%), s510 (61.2%) and sand (54.4%). The power improvement could be observed only in some of the circuits, which are having low boundary state probabilities. This happens because in our GA based approach we have targeted a two-level realization. On the otherhand, JEDI targets a multi-level realization. The Synopsys tool also generates a multi-level implementation. Thus, it is expected that JEDI output will be more amenable for synthesis using Synopsys tool.

We have also presented the combinational area (which is the number of the product terms) comparison of the power-gating approach in Table 4(a) with the area needed for NOVA [27] encoding and GA encoding without power-gating. While the GA approach without power-gating requires on an average 8.7% extra product terms over NOVA, the power-gating approach needs about 10% extra product terms. In Table 4(b), we have compared the area of the power-gating approach with the multilevel state assignment tool JEDI in 90 nm technology using Synopsys synthesis tool. In this technology, a 2-input NAND gate has an area of 15.5%.
Next, in Tables 5 and 6, we present the results of the new power model explained in Section 4. Here, we have considered the combinational and sequential power of the resulting circuit. Table 5 gives the dynamic power results of the circuits using our power model. dyn0 and dyn1 are the combinational switching activities of the first and second machine, respectively. tot_dyn is the total switching activity of the machine calculated using Eq. (1). Nova_dyn is the total switching activity of the combinational part of the original machine after encoding using NOVA.Nova_dyn and Nova_seq_dyn are the dynamic powers calculated using Eq. (1), after our encoding and NOVA encoding of the machine respectively. The result shows on an average 38.5% savings in dynamic power. The maximum savings in dynamic power are for circuits like s208 (76%), s832 (75.1%) and s208 (69.6%). Saving in dynamic power is higher for the cases with lower boundary state probabilities. For FSMs with higher boundary state probabilities, both the machines are always on, thus requiring higher power. Circuits, such as s208, s832 and s208 have lower boundary state probabilities, so lower power dissipation. Leak0 and Leak1 are the combinational leakage power of the first and second machine, estimated using our leakage model given in Section 4.2 of the final encoded circuit. Total leakage is calculated using Eq. (1) and is given in the column tot_leak of Table 6. This leakage is in µW. Nova_Leak (µW) is the leakage power of the combinational part of the machine after encoding using NOVA. Nova_Leak and Nova_seq_Leak are the leakage results of the sequential part of the machine, calculated using Eq. (5), after our encoding and NOVA encoding respectively. The result shows on an average 9% saving in leakage power. The maximum savings in leakage power are for circuits like s820 (87.1%), s832 (87.1%).

Table 4

<table>
<thead>
<tr>
<th>Circuit</th>
<th>NOVA</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beecount</td>
<td>323</td>
<td>389</td>
</tr>
<tr>
<td>dk14</td>
<td>318</td>
<td>493</td>
</tr>
<tr>
<td>dk16</td>
<td>902</td>
<td>861</td>
</tr>
<tr>
<td>dk512</td>
<td>291</td>
<td>294</td>
</tr>
<tr>
<td>dk27</td>
<td>126</td>
<td>136</td>
</tr>
<tr>
<td>Donfile</td>
<td>530</td>
<td>580</td>
</tr>
<tr>
<td>ex4</td>
<td>281</td>
<td>331</td>
</tr>
<tr>
<td>Sand</td>
<td>1386</td>
<td>1096</td>
</tr>
<tr>
<td>s1</td>
<td>805</td>
<td>405</td>
</tr>
<tr>
<td>s208</td>
<td>431</td>
<td>480</td>
</tr>
<tr>
<td>s386</td>
<td>548</td>
<td>540</td>
</tr>
<tr>
<td>s510</td>
<td>1027</td>
<td>908</td>
</tr>
<tr>
<td>s820</td>
<td>1569</td>
<td>923</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>Circuit</th>
<th>dyn0</th>
<th>dyn1</th>
<th>tot_dyn</th>
<th>Nova_dyn</th>
<th>Seq_dyn</th>
<th>Nova_seq_dyn</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbara</td>
<td>695.9</td>
<td>232.9</td>
<td>784.22</td>
<td>921.71</td>
<td>1.71</td>
<td>1.71</td>
</tr>
<tr>
<td>bbtas</td>
<td>76.04</td>
<td>81.62</td>
<td>139.79</td>
<td>120.04</td>
<td>1.35</td>
<td>1.37</td>
</tr>
<tr>
<td>dk14</td>
<td>447.95</td>
<td>642.1</td>
<td>1090.05</td>
<td>739</td>
<td>1.37</td>
<td>1.52</td>
</tr>
<tr>
<td>dk16</td>
<td>505.2</td>
<td>515.3</td>
<td>1020.5</td>
<td>1036.28</td>
<td>2.48</td>
<td>2.51</td>
</tr>
<tr>
<td>dk512</td>
<td>113.56</td>
<td>112.08</td>
<td>187.42</td>
<td>216.17</td>
<td>2.22</td>
<td>2.05</td>
</tr>
<tr>
<td>dk27</td>
<td>54.04</td>
<td>53.86</td>
<td>107.9</td>
<td>103.34</td>
<td>1.6</td>
<td>1.52</td>
</tr>
<tr>
<td>Donfile</td>
<td>348</td>
<td>335</td>
<td>683</td>
<td>800.5</td>
<td>2.4</td>
<td>2.36</td>
</tr>
<tr>
<td>ex4</td>
<td>158.4</td>
<td>272.6</td>
<td>340.05</td>
<td>345.05</td>
<td>2.4</td>
<td>2.36</td>
</tr>
<tr>
<td>sand</td>
<td>3748</td>
<td>262.2</td>
<td>2454.4</td>
<td>3966.2</td>
<td>2.3</td>
<td>2.23</td>
</tr>
<tr>
<td>s1</td>
<td>1839.9</td>
<td>373.89</td>
<td>2163.39</td>
<td>2873.58</td>
<td>2.3</td>
<td>2.23</td>
</tr>
<tr>
<td>s208</td>
<td>204.2</td>
<td>531.4</td>
<td>237.07</td>
<td>136.08</td>
<td>2.15</td>
<td>2.23</td>
</tr>
<tr>
<td>s386</td>
<td>0.604</td>
<td>1.068</td>
<td>0.67</td>
<td>2.18</td>
<td>0.79</td>
<td>0.688</td>
</tr>
<tr>
<td>s510</td>
<td>1.171</td>
<td>0.483</td>
<td>1.29</td>
<td>3.21</td>
<td>1.02</td>
<td>1.35</td>
</tr>
<tr>
<td>s820</td>
<td>0.977</td>
<td>0.71</td>
<td>1.005</td>
<td>4.2</td>
<td>0.559</td>
<td>0.919</td>
</tr>
<tr>
<td>s382</td>
<td>0.98</td>
<td>0.567</td>
<td>4.14</td>
<td>0.559</td>
<td>0.87</td>
<td></td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>Circuit</th>
<th>POWER GATED_AREA</th>
<th>JEDI_AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbtas</td>
<td>34</td>
<td>26</td>
</tr>
<tr>
<td>bbara</td>
<td>34</td>
<td>26</td>
</tr>
<tr>
<td>dk14</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td>dk16</td>
<td>65</td>
<td>62</td>
</tr>
<tr>
<td>dk512</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>dk27</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Donfile</td>
<td>39</td>
<td>47</td>
</tr>
<tr>
<td>ex4</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>moduolo12</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>sand</td>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>s820</td>
<td>76</td>
<td>66</td>
</tr>
<tr>
<td>s510</td>
<td>71</td>
<td>66</td>
</tr>
<tr>
<td>s832</td>
<td>81</td>
<td>64</td>
</tr>
</tbody>
</table>

5.1. Impact on performance:

While comparing the delay of the circuits synthesized by our approach with the NOVA encoded circuits, the difference found is not significant. This happens due to the following reason. From
Our approach has three components. As well (as noted in blocks COMB1 and COMB2 are expected to be simpler than in the gible. The output multiplexers have a delay of 41.7 ps at 90 nm directly determines the LE. Hence, the associated delay is negligible. The output multiplexers have a delay of 41.7 ps at 90 nm. 

6. Conclusion

We have presented an efficient technique for synthesizing FSMs using power-gating targeting total power saving. The idea of combined partitioning and state encoding is introduced for the first time in the synthesis process in the genetic algorithm formulation. The technique worked well as verified by the experimentation with a number of benchmark circuits. The benefit of the scheme presented in the work, is best taken by the circuits having lesser number of boundary states.

References