

Leakage Current Reduction in CMOS VLSI Circuits by Input Vector Control

Afshin Abdollahi, Farzan Fallah, and Massoud Pedram

Abstract—The first part of this paper describes two runtime mechanisms for reducing the leakage current of a CMOS circuit. In both cases, it is assumed that the system or environment produces a “sleep” signal that can be used to indicate that the circuit is in a standby mode. In the first method, the “sleep” signal is used to shift in a new set of external inputs and pre-selected internal signals into the circuit with the goal of setting the logic values of all of the internal signals so as to minimize the total leakage current in the circuit. This minimization is possible because the leakage current of a CMOS gate is strongly dependent on the input combination applied to its inputs. In the second method, nMOS and pMOS transistors are added to some of the gates in the circuit to increase the controllability of the internal signals of the circuit and decrease the leakage current of the gates using the “stack effect”. This is, however, done carefully so that the minimum leakage is achieved subject to a delay constraint for all input–output paths in the circuit. In both cases, Boolean satisfiability is used to formulate the problems, which are subsequently solved by employing a highly efficient SAT solver. Experimental results on the combinational circuits in the MCNC91 benchmark suite demonstrate that it is possible to reduce the leakage current in combinational circuits by an average of 25% with only a 5% delay penalty. The second part of this paper presents a design technique for applying the minimum leakage input to a sequential circuit. The proposed method uses the built-in scan-chains in a VLSI circuit to drive it with the minimum leakage vector when it enters the sleep mode. The use of these scan registers eliminates the area and delay overhead of the additional circuitry that would otherwise be needed to apply the minimum leakage vector to the circuit. Experimental results on the sequential circuits in the MCNC91 benchmark suit show that, by using the proposed method, it is possible to reduce the leakage by an average of 25% with practically no delay penalty.

Index Terms—Leakage current control, low power design, minimum leakage vector, scan chain, VLSI circuits.

I. INTRODUCTION

THE rapid increase in the number of transistors on chips has enabled a dramatic increase in the performance of computing systems. However, the performance improvement has been accompanied by an increase in power dissipation; thus, requiring more expensive packaging and cooling technology. Historically, the primary contributor to power dissipation in CMOS circuits has been the charging and discharging of load capacitances, often referred to as the dynamic power dissipation. This component of power dissipation is quadratically proportional to the supply voltage level. Therefore, in the past, chip designers

have relied on scaling down the supply voltage to reduce the dynamic power dissipation. Maintaining the transistor switching speeds requires a proportionate downscaling of the transistor threshold voltages in lock step with the supply voltage reduction. However, threshold voltage scaling results in a significant amount of leakage power dissipation due to an exponential increase in the subthreshold leakage current conduction. Borkar in [1] predicts a 7.5-fold increase in the leakage current and a five-fold increase in total energy dissipation for every new microprocessor chip generation.

There are three main sources for leakage current:

- 1) source/drain junction leakage current;
- 2) gate direct tunneling leakage;
- 3) subthreshold leakage through the channel of an OFF transistor.

The junction leakage occurs from the source or drain to the substrate through the reverse-biased diodes when a transistor is OFF. The magnitude of the diode’s leakage current depends on the area of the drain diffusion and the leakage current density, which, in turn, is determined by the process technology.

The gate direct tunneling leakage flows from the gate through the “leaky” oxide insulation to the substrate. Its magnitude increases exponentially with the gate oxide thickness T_{ox} and supply voltage V_{DD} . According to the 2001 International Technology Roadmap for Semiconductors, high-K gate dielectric reduced direct tunneling current is required to control this component of the leakage current for low standby power devices.

The subthreshold current is the drain-source current of an OFF transistor. This is due to the diffusion current of the minority carriers in the channel for a MOS device operating in the weak inversion mode (i.e., the subthreshold region.) For instance, in the case of an inverter with a low input voltage, the nMOS is turned OFF and the output voltage is high. Even when V_{GS} is 0V, there is still a current flowing in the channel of the OFF nMOS transistor due to the VDD potential of the VDS. The magnitude of the subthreshold current is a function of the temperature, supply voltage, device size, and the process parameters, out of which, the threshold voltage (V_{th}) plays a dominant role.

In current CMOS technologies, the subthreshold leakage current is much larger than the other leakage current components. This current can be calculated by using the following equation:

$$I_{DS} = K \left(1 - e^{-V_{DS}/V_T} \right) e^{(V_{GS} - V_T + \eta V_{DS})/nV_T}$$

where K and n are functions of the technology, and η is the drain-induced barrier lowering coefficient. Clearly, decreasing the threshold voltage increases the leakage current exponentially. In fact decreasing the threshold voltage by 100-mv

Manuscript received September 13, 2003.

A. Abdollahi and M. Pedram are with the University of Southern California, Los Angeles, CA 90089 USA (e-mail: afshin@usc.edu; pedram@ceng.usc.edu).

F. Fallah is with the Fujitsu Laboratories of America, San Jose, CA 94085 USA (e-mail: farzan@fla.fujitsu.com).

Digital Object Identifier 10.1109/TVLSI.2003.821546

increases the leakage current by a factor of 10. Decreasing the length of transistors increases the leakage current as well. Therefore, in a chip, transistors that have smaller threshold voltage and/or length due to process variation, contribute more to the overall leakage. Although previously the leakage current was important only in systems with long inactive periods (e.g., pagers and networks of sensors), it has become a critical design concern in any system in today's designs.

Unlike the dynamic power, which depends on the average number of switching transistors per clock cycle, the leakage power depends on the number of on-chip transistors, regardless of their average switching activity. The input pattern dependence of the leakage current makes the problem of determining the leakage power dissipated by a circuit a difficult one. This statement is true even when runtime statistics about the active versus idle times for a circuit are known. This is because by applying the minimum-leakage producing input combination to the circuit when it is in the idle mode, we can significantly reduce the leakage power dissipation of the circuit. Consequently, identification of a minimum leakage vector (MLV) is an important problem in low power design of VLSI circuits.

In this paper, several runtime mechanisms for leakage current reduction of CMOS VLSI are introduced. Our methods find the MLV of a circuit and the optimum way of modifying the circuit to reduce its leakage current using a Boolean satisfiability formulation. Our proposed technique is applicable to both combinational and sequential circuits. For the latter type of circuits, our method requires only modification of the scan-chains that are already put into the circuit in order to allow efficient testing of the circuit functionality. No other change to the circuit in question is required. So from a designer's perspective, the cost of reducing leakage in a standby circuit is minimal. Parts of this archival paper have appeared in [2], [3].

In Section II, a review of a number of the leakage reduction techniques is presented. In Section III, we describe a method for finding the MLV and its corresponding leakage current. Our method is based on constructing a Boolean network for computing the leakage current of a VLSI circuit and solving a series of Boolean satisfiability problems corresponding to that network. We use an incremental satisfiability solver technique to speedup the operation [4]. We minimize the leakage current by using an MLV to drive the circuit while in the standby mode. In Section IV, two improved mechanisms for leakage current reduction are introduced. The basic idea is to increase the controllability of the internal signals of a circuit. Using multiplexers or modifying the internal gates of the circuit achieves this. Experimental results for combinational circuits are presented in Section V. In Section VI, scan-based testing is described. Our method for modifying the scan-chain of a sequential circuit to decrease its leakage current is presented in Section VII. Experimental results for sequential circuits are presented in Section VIII. Finally, we conclude the paper in Section IX.

II. PREVIOUS WORK

In this section, we briefly review a number of commonly used leakage reduction techniques.

A. Leakage Reduction by Input Vector Control

Many researchers have used models and algorithms to estimate the nominal leakage current of a circuit [5]–[7]. The minimum and maximum leakage currents of a circuit have been estimated using a greedy heuristic in [8]. Because of the transistor stacking effect, the leakage of a circuit depends on its input combination [8]. Table I shows different leakage values for all input combinations of a 3-input *NAND* gate.

As can be seen the leakage current ratio between different input combinations can be as high as 10. As the operational state of the transistors that constitute a CMOS gate are determined by their input signal values, the goal can be expressed as finding the input pattern that maximizes the number of disabled (off) transistors in all stacks across the circuit [9]. The authors in [10] provided an estimation of the maximum leakage current by greedily assigning input combinations of logic blocks that result in high leakage currents. All the methods above can be used to determine the minimum-leakage vector and to further exploit the stacking effect by inserting transistors in the leaky sections of a circuit [11]. Another possibility is to perform an exhaustive circuit-level simulation for all input patterns to find the pattern with the minimum leakage current. However, this approach is not practical for large circuits. In [12], the authors used probabilistic methods to reduce the number of simulations necessary to find a solution with a desired accuracy. In [2], [13], a SAT-based formulation for finding the minimum leakage vector at the circuit inputs is described. No circuit modification was proposed in [13]. In contrast, in [2], the authors introduced a method for controlling the internal nodes by modifying some gates and without using extra multiplexers. Moreover, in [2], delay constraints are explicitly accounted for and the optimal subset of internal nodes of the circuit to be controlled is determined by the SAT formulation. Having found the minimum leakage pattern, one can use this vector to drive the circuit while in standby mode. This requires the addition of a number of multiplexers at the primary inputs of the circuit. The multiplexers are controlled using a sleep signal. Because the power reduction using this technique can be achieved only for long sleep periods, a threshold is used to activate the sleep signal only if the sleep period is long enough.

B. Leakage Reduction by Increasing the Threshold Voltages

One way of decreasing the leakage current is increasing the threshold voltages of transistors. There are several ways to do this, but in all of them some process technology modification is necessary. However, this may not always be possible. Another approach is to use high-threshold voltage devices on non-critical paths so as to reduce the leakage power while using low-threshold devices on critical paths so that the circuit performance is maintained. This technique requires an algorithm that searches for the gates where the high-threshold voltage devices can be used [14]. This technique has been called the Dual V_{th} CMOS. In Dynamic Threshold MOS (DTMOS), the body and the gate of each transistor are tied together such that when the device is off, the leakage is low. If the device is on, then the current will be high [15]. Among the techniques that dynamically modify the threshold voltage during runtime, the classic

TABLE I
LEAKAGE CURRENT VALUES FOR DIFFERENT INPUT COMBINATIONS OF
A 3-INPUT NAND GATE

state	leakage
000	3.21×10^{-12}
001	4.62×10^{-12}
010	4.67×10^{-12}
011	3.13×10^{-11}
100	6.47×10^{-12}
101	3.21×10^{-11}
110	3.67×10^{-11}
111	3.23×10^{-1}

example is standby power reduction (SPR) or variable threshold CMOS (VTCMOS). In this method V_{th} is raised during the standby mode by making the substrate voltage either higher than V_{dd} (for p transistors) or lower than ground (for N transistors). However, this technique requires an additional power supply, which may not be attractive in some commercial designs. A technique presented in [16] successfully solves this problem and applies the technique to a commercial digital signal processor. The architectural support needed to use VTCMOS can be done in hardware or software. There is a large performance penalty due to the time required removing the substrate voltage to return to the normal operation mode. Noise immunity problems have been reported when the substrate voltage is changed, but since, in this case the technique is applied when the system is idle, there is no negative effect on the normal operation of the circuit.

C. Leakage Reduction by Gating the Supply Voltage

The last approach considered is power supply gating. There are many ways in which this technique can be implemented, but the basic idea is to shut down the power supply so the idle units do not consume any power. This can be done using some high threshold transistors called sleep transistors [17]. If the threshold voltages of sleep transistors are changed at runtime, the triple-well technology is required. Another possibility is to use multiple-threshold voltage CMOS (MTCMOS) [18]. In MTCMOS, a high threshold device is inserted in the series with low threshold transistors creating a sleep transistor. This creates virtual supply and ground rails whose voltage levels are very close to the real supply and ground lines because of the very small on-resistance of the inserted high- V_{th} transistors. In practice, only one virtual rail (usually the virtual ground) is used. Normally, one sleep transistor per gate is used, but larger granularities are possible, which require fewer transistors. The problems with this technique are reduced performance and noise immunity.

III. LEAKAGE MINIMIZATION BY INPUT VECTOR CONTROL

By applying a MLV to a circuit, it is possible to decrease the leakage current of the circuit when it is in the standby mode. Note that applying MLV for leakage reduction is independent of the source of leakage, which may include the subthreshold

and the gate tunneling leakage currents. For our experimental results, we have used SPICE to measure the leakage current of different gates under various input combinations. SPICE simulator reports a leakage current value that includes both the sub-threshold leakage and the gate leakage currents.

We assume that the environment in which the circuit is placed e.g., with the aid of a power management unit, generates a *SLEEP* signal for the circuit. This signal is then used to initiate the application of the MLV to the circuit inputs. To use this method for leakage reduction, it is necessary to find an input vector that causes the minimum leakage current in a VLSI circuit. A *trivial lower (upper) bound* on the leakage current is the sum of the minimum (maximum) leakage currents of all logic gates in the circuit. However, this may not correspond to any feasible solution because the input combination that produces the minimum (maximum) leakage in some gate, $gate_i$, may conflict with the one that produces the minimum leakage for another gate, $gate_j$. In the remainder of this section, we describe an algorithm for finding an MLV for a given combinational logic circuit. More precisely, given a combinational logic circuit description, we first construct a Boolean network, which computes the total leakage of that circuit. We call the resulting circuit a *Leakage Computing Network (LCN)*. Next from the LCN description, we write a set of Boolean clauses that capture the leakage current of the original circuit. We employ a SAT solver to find an input vector that results in a leakage less than a given number C . Next, we perform a linear search on the value of C to find the MLV. Finally, we modify the original circuit by adding a number of multiplexers to shift in the MLV when the circuit enters the idle mode. Notice that the LCN is only used as a computational tool and the only actual hardware is the original circuit and the final circuit (which is augmented by the multiplexers and the MLV vector). The leakage current of a logic gate depends on its input values. Let $leakage(X_j)$ be the leakage current of the j th gate of a circuit under the immediate input vector combination \bar{X}_j . Notice that $leakage(X)_j$ can be written as a sum of up to 2^n terms, where n is the number of inputs of the gate. For example, the following equation gives the leakage current for all input values of a two-input NAND gate:

$$leakage(X_j) = \bar{X}_{j1}\bar{X}_{j0}L_{00} + \bar{X}_{j1}X_{j0}L_{01} \\ + X_{j1}\bar{X}_{j0}L_{10} + X_{j1}X_{j0}L_{11}$$

where L_{pq} is the leakage current of the gate when $X_{j1} = p$ and $X_{j0} = q$. Without loss of generality, we multiply all gate leakage values with a large constant number to make them integer values. The leakage current minimization problem can then be stated as follows:

Given circuit-induced logic dependencies among X_j 's, find a primary input vector that minimizes $\sum_j leakage(X_j)$ for all gates in the circuit.

The cost function above can be directly implemented in the LCN by using adders and multiplexers. However, to decrease the number of adders, we use the following approach. First we compute the sum of all cost function terms that correspond to some leakage value L_{kl} . Next we compute the sum of results. As an example, consider a circuit with two NAND gates, denoted

by $gate_i$ and $gate_j$. In a straightforward LCN realization, the following sum is computed:

$$\begin{aligned} & Leakage(X_i) + Leakage(X_j) \\ &= (\bar{X}_{i1}\bar{X}_{i0}L_{00} + \bar{X}_{i1}X_{i0}L_{01} + X_{i1}\bar{X}_{i0}L_{10} + X_{i1}X_{i0}L_{11} \\ &+ (\bar{X}_{j1}\bar{X}_{j0}L_{00} + \bar{X}_{j1}X_{j0}L_{01} + X_{j1}\bar{X}_{j0}L_{10} + X_{j1}X_{j0}L_{11})) \end{aligned}$$

where X_i is a Boolean variable and L_{ij} is a fixed-length vector of Boolean variables corresponding to the binary representation of the actual leakage value. The LCN size can be reduced if we rearrange the terms as follows:

$$\begin{aligned} & Leakage(X_i) + Leakage(X_j) \\ &= (\bar{X}_{i1}\bar{X}_{i0} + \bar{X}_{j1}X_{j0})L_{00} \\ &+ (\bar{X}_{i1}X_{i0} + \bar{X}_{j1}X_{j0})L_{01} \\ &+ (\bar{X}_{i1}\bar{X}_{i0} + X_{j1}\bar{X}_{j0})L_{10} \\ &+ (X_{i1}X_{i0} + X_{j1}X_{j0})L_{11}. \end{aligned}$$

The reason is that in the latter case, for each leakage value, instead of computing the sum of n terms each with m bits, we compute the sum of n single-bit numbers. Then, multiply the result with an m -bit number. The first approach needs $m(n-1)$ single-bit adders, while the second requires $n-1 + m \log n$ single-bit adders. Thus, the second approach is more efficient. To compute the total leakage in our approach, we use a decoder for each gate. As an example consider a 2-input gate with four different leakage values corresponding to four different combinations of its inputs. Fig. 1 shows a 2-to-4 decoder associated with this gate in the LCN. In this figure, D_{ij}^k values represent the input combination ij of $gate_k$.

Fig. 2 shows the LCN structure for computing the total leakage current of all gates of type k in the original circuit that performs the same Boolean operation (e.g., two-input NAND). The one's counters in this figure count the number of D_{ij}^k variables that are assigned a value of ONE. For example, if there are 50 two-input NAND gates and 20 of them receive input combination 00, while 15, 10, and 5 gates receive 01, 11, and 10 input combinations, respectively, then the total leakage of all two-input NAND gates in the circuit will be $20L_{00} + 15L_{01} + 10L_{11} + 5L_{10}$.

Notice that when the leakage current of a gate type for a specific input combination is equal to that of another gate for some other input combination, it is possible to share the logic structures between them to improve the size efficiency of the LCN. The total leakage current of the circuit is computed by summing up all LT_k values corresponding to all gate types in the original circuit. Suppose we are interested in finding a vector whose quantized leakage current is less than a given leakage l which is quantized to the integer number C . (for example if LB and UB are lower bound and upper bound on leakage and we intend to compare the leakage of the circuit with l then C is defined as $C = \lfloor l - LB/UB - LBn \rfloor$ where n is the desired number of quantized levels.) To do this, we compare the total circuit leakage with C . Fig. 3 shows the circuit realization for comparing the total circuit leakage with C .

We model the circuit in Fig. 3 using Boolean clauses as described in [19].

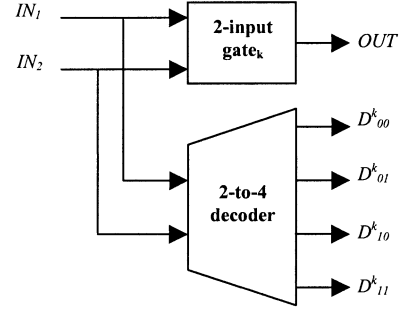


Fig. 1. A 2-to-4 decoder indicating input combinations of a 2-input logic gate.

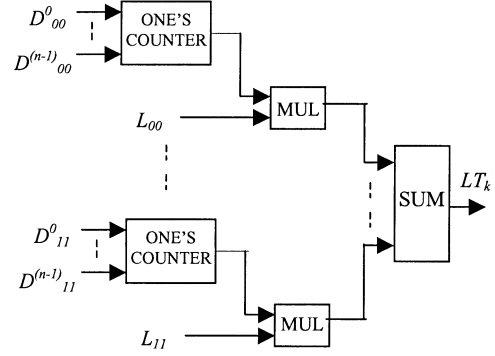


Fig. 2. Contribution of all gates of type k to the total leakage.

For example, if $n = 2$, $LT_1 = [a_1 a_0]$ and $LT_2 = [b_1 b_0]$ then the summation of these two vectors is $L_{total} = [s_2 s_1 s_0]$. The Boolean description of the relation between a_0 , b_0 , and s_0 is $s_0 = XOR(a_0, b_0)$ and this Boolean relation can be described by four clauses: $Clause_1 = a_0 + b_0 + \bar{s}_0$, $Clause_2 = a_0 + \bar{b}_0 + s_0$, $Clause_3 = \bar{a}_0 + b_0 + s_0$, and $Clause_4 = \bar{a}_0 + \bar{b}_0 + \bar{s}_0$. The space complexity of the SAT problem formulation is linear in the size of the original Boolean network. The time complexity of the SAT solver is exponential in the worst case.

Algorithm LIN_SEARCH_FOR_MLV:

- 1) Find the trivial bounds on leakage current, LB and UB described in the beginning of Section III
- 2) $C = UB$, $mlv = \{\}$
- 3) Write Boolean clauses to model the circuit leakage and the condition that $total_leakage \leq C$
- 4) Solve the resulting SAT problem
- 5) If there is no solution, stop; $C + 1$ is the minimum leakage and mlv is the solution
- 6) $mlv =$ the vector found by the SAT solver
- 7) $C = C - 1$
- 8) If $C < LB$, stop; $C + 1$ is the minimum leakage and mlv is the solution
- 9) Go to step 3

The above algorithm performs a linear search on the values between LB and UB to find the minimum leakage current. The

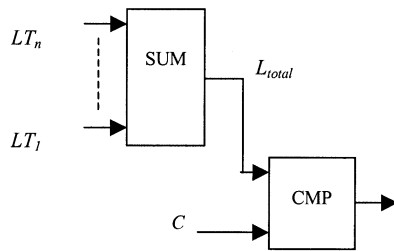


Fig. 3. Comparing circuit leakage with C .

search starts from UB and proceeds toward LB . During the search all problems are feasible except the last one. Note that the constraints corresponding to $totalLeakage \leq C - 1$ are tighter than the ones corresponding to $totalLeakage \leq C$. Thus, every solution of iteration $i + 1$ is a solution of iteration i . In every iteration, the SAT solver produces many conflict clauses during the search for the answer.¹ We use this fact to speedup the search by using the conflict clauses that are generated during the i th iteration and adding new clauses to them to model the $(i + 1)$ th iteration. This is instrumental in substantially decreasing the computation time.

It is possible to start the search from LB toward UB . In this case all problems except the last one are infeasible. Because this formulation does not permit the reuse of the conflict clauses, it is slower than the one described previously. A binary search, rather than a linear search may also be used. Again we note that a binary search does not permit the reuse of the conflict clauses. Furthermore, the decrease in the number of iterations (sub-problems) tends to be very small compared to the linear search. Therefore, using a linear search algorithm provides the best runtime. After finding the MLV, we use it to drive the circuit every time the $SLEEP$ signal is activated. This can be accomplished by using some multiplexers controlled by the $SLEEP$ signal to drive the inputs of the circuit. Simplifying the multiplexers based on the fact that one input of each multiplexer is a constant 0 or 1 reduces the hardware cost. Fig. 4 shows the input driver for two bits $\{a_1, a_0\}$ assuming the min leakage vector is $\{1, 0\}$.

IV. LEAKAGE REDUCTION BY ADDING CONTROL POINTS

In Section II, we reduced the leakage current by using an input vector control mechanism. However, in circuits with large logic depth, an externally applied input vector may effectively control only the gates that are close to primary inputs. If we find a way to directly control at least some of the internal nodes of a circuit, we can further reduce the leakage of the circuit. In this section we introduce two methods to add control points to a circuit to decrease its leakage.

A. Using Multiplexers

An easy way to control the value of an internal signal (line) of a circuit is to cut the internal line and insert a 2-to-1 multi-

¹Conflict arises when during the search one or more clauses become unsatisfiable in the current search sub space. The SAT algorithm backtracks from this point and also learns from the conflict by adding one or more conflict clauses to its database. Adding such conflict clauses prevents the algorithm from encountering the same conflict. In other words, clauses prune the search space efficiently [20].

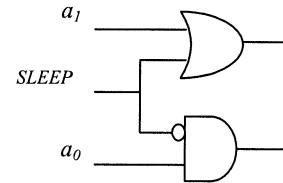


Fig. 4. Input driver for min leakage vector $\{1, 0\}$.

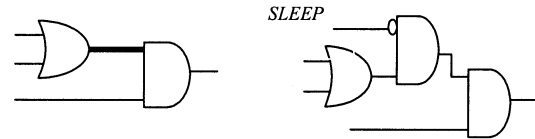


Fig. 5. Replacing a line by an AND gate.

plexer that is controlled by the $SLEEP$ signal. The two inputs of the multiplexer include the incoming signal and a $ZERO$ or ONE value decided by the leakage current minimization algorithm. The output is the outgoing signal. Since one input of the multiplexer is a fixed value, instead of the multiplexer, an AND gate or an OR gate may be used. Fig. 5 shows a part of a circuit before and after replacing its internal line by an AND gate.

In Fig. 5(b), in the sleep mode, the output of the AND gate is $ZERO$; if, based on the result of leakage current minimization algorithm, we need to have a ONE on that line in the idled circuit, the AND gate has to be replaced by an OR gate. The additional AND or OR gate and the gates in its fanout consume dynamic power when a new value is shifted into the circuit at the beginning and the end of the circuit idle time. In addition, the additional gates consume dynamic power when the circuit is in the running mode. However, this dynamic power consumption overhead is negligible if the idle time is sufficiently long. We assume that the power management unit for the whole design knows about this overhead and will only activate the $SLEEP$ signal if the idle time is expected to be very long. In this paper, we do not concern ourselves with how such a global power management policy for a complete design can be developed and put in place.

When a new control gate is added to the circuit, there will also be an additional leakage current associated with that gate. The algorithm that determines the number, type, and insertion location of the control gates inside a combinational logic block must account for the leakage currents of these gates. In the remainder of this subsection we present a method to optimally select a subset of the internal lines in a circuit to be replaced with AND or OR gates. The method is based on modifying the LCN by adding additional input variables corresponding to the internal lines of the circuit. In other words, for each internal line in the circuit, two new variables X and Y are introduced. The value of X determines whether or not the connection will be replaced by a multiplexer. If $X = 1$, then a multiplexer whose inputs are the original line and a variable Y , is inserted on that line. The LCN is modified to account for the leakage of the added gate (cf. Fig. 6.)

Now the problem of minimizing the leakage current can be described as minimizing the value of L'_{total} which is a function of input vector and also variables X 's and Y 's. By running

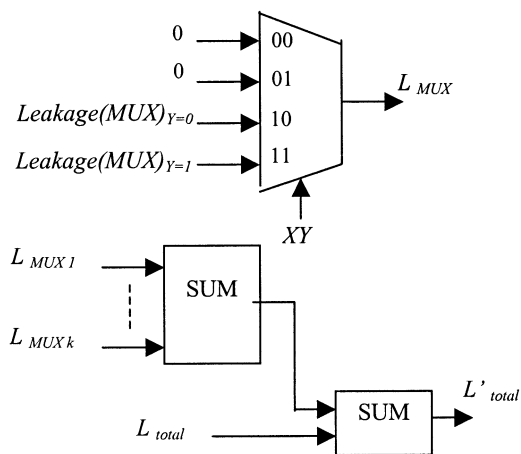


Fig. 6. Adding the leakage current of the multiplexer to the total leakage.

LIN_SEARCH_FOR_MLV on the modified LCN with extra variables (X 's and Y 's), we can obtain the following:

- 1) MLV;
- 2) Internal lines on which multiplexers are inserted;
- 3) Y value for each multiplexer and customization of the multiplexer to an *AND* or *OR* gate based on the Y value.

Our minimization algorithm finds the optimum subset of internal lines on which multiplexers are inserted. The minimization algorithm considers the advantage of controlling the internal lines in the circuit and weighs it against the disadvantage of additional leakage current due to the required multiplexers. Since the minimization algorithm searches for the minimum leakage solution, if adding any multiplexer helps decrease the leakage, it will be added to the circuit.

B. Modifying Gates

The leakage cost of multiplexers serves as a disincentive to employ a large number of these multiplexers in the circuit. In this subsection we propose an alternative method to control the outputs of internal gates in a circuit. Since the new method does not add any gate to the circuit, there is no extra leakage associated with adding a control point to the circuit.

We use two variables X and Y for each gate in the circuit. The value of X determines whether or not a gate in the circuit undergoes some change. The value of Y determines the way that the gate is changed. Consider a fully-complementary CMOS gate, $out = g(in)$. Based on the values of X and Y , which are in turn computed by our leakage minimization algorithm; This gate is changed as follows:

```

If ( $X == 0$ )  $out = g(in)$ 
else
if ( $Y == 0$ )  $out = AND(NOT(SLEEP), g(in))$ 
else  $out = OR(SLEEP, g(in))$ 
    
```

As described above, modifying this gate enables controlling the output of the gate independent of its inputs in the standby mode. In other words, if we must have a *ONE* at the output of the gate when in the standby mode, we replace the gate with

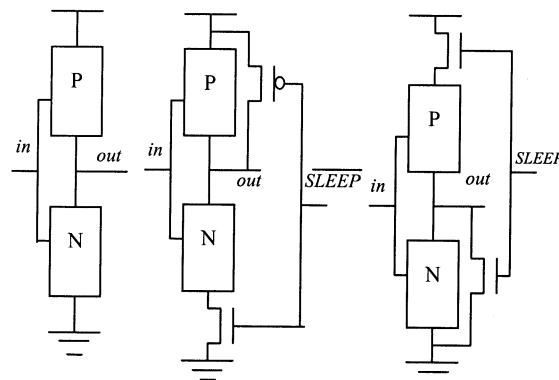


Fig. 7. From left to right, a fully-complementary CMOS gate implementing $out = g(in)$, a modified gate realizing $out = OR(SLEEP, g(in))$, and another gate realizing $out = AND(NOT(SLEEP), g(in))$.

$OR(SLEEP, g(in))$. Similarly, if we ought to have a *ZERO*, we replace it with $AND(NOT(SLEEP), g(in))$.

Fig. 7 shows a CMOS gate with its pMOS and nMOS sections and two ways to modify the gate. Note that in both cases a transistor is added in the series with one of the N or P sections, which is not similar to adding sleep transistors to the circuit. Adding sleep transistors with possibly different threshold voltages or sizes to the gates may result in a large number of complications, including the unreasonable delay and leakage peaks while waking up a “sleeping” gate, or “shutting” down a gate. However, we do not add any sleep transistors to the gates. Instead we simply replace the original gate with another logic gate with an extra input (*sleep*) and identical functionality when $sleep = 0$.

The percentage of the reduction depends on the original number of transistors in the gate [11]. Moreover, as mentioned before, this method enables us to control the values of the internal lines in the circuit; thus, reducing the leakage current of the gates in the fanout of the lines. Modifying a gate in this way results in a delay and an area penalty. For example, in case B, the high-to-low transition becomes slower, whereas in case C the low-to-high propagation delay is increased. We take the *pin-dependent* propagation delay of a gate to be the average of input-output gate delays for the rising and falling transitions. Obviously, the delay and area penalties depend on the sizes of the added transistors in each case. We size these transistors so that the increase in the delay and the area of each gate is no more than some percentage (Section V).

In the sequel, we present a method to extend the LCN so that the leakage minimization is performed subject to a delay constraint on all of the primary input to primary output paths in the circuit. The left circuit structure in Fig. 8 selects the correct value of the leakage for each gate in the circuit whereas the right structure does the same for delay calculation.

Note in this figure $leakage_A$ and $delay_A$ denote the leakage current and propagation delay of the gate without modification (i.e., $out = g(in)$). $leakage_B$ and $delay_B$ denote the leakage and propagation delay of the gate modified to $out = OR(SLEEP, g(in))$. $leakage_C$ and $delay_C$ indicate the same for the case where $out = AND(NOT(SLEEP), g(in))$. As in static timing analysis, the gate delay values are used to calculate the maximum delay of the circuit for all input-output paths using the circuit shown in Fig. 9. The arrival time of each gate is

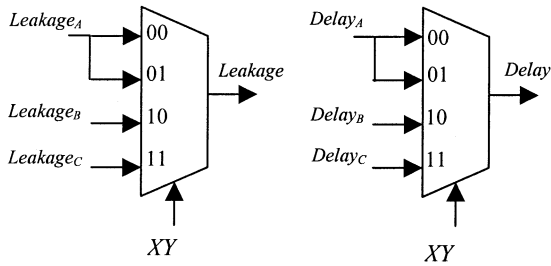


Fig. 8. Leakage and delay values of a modified gate.

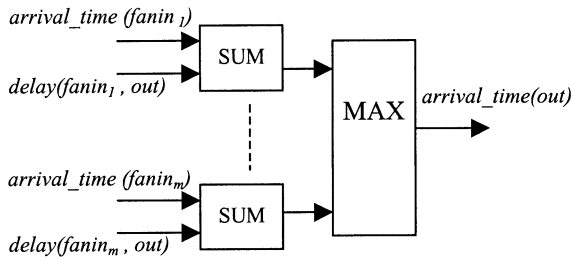


Fig. 9. Calculating the output arrival time of a gate.

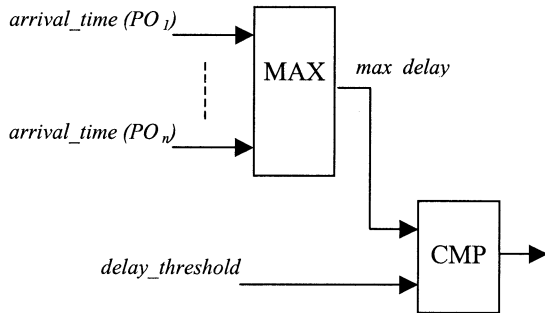


Fig. 10. Comparing the maximum delay of the circuit with a delay threshold.

the maximum of the sum of the arrival time of each of its inputs and the pin-dependent delay from that input to the output of the gate.

The maximum delay of the circuit is the maximum of arrival times of its primary outputs. Fig. 10 shows the circuit for comparing the maximum delay of the circuit with a given threshold.

The leakage minimization problem can be stated as that of minimizing the value of L_{total} which is a function of input vector and also variables X 's and Y 's. The leakage minimization has to be performed under the delay constraint illustrated in Fig. 10. Therefore, the minimization algorithm should take into account the values of the output of both circuits in Figs. 3 and 10 as depicted in Fig. 11.

By running LIN_SEARCH_FOR_MLV on the modified LCN with the aforementioned Delay Computing Network (DLN) and variables (X 's and Y 's), we can obtain the following:

- 1) MLV;
- 2) Gates that are structurally modified;
- 3) Y value for each modified gate, which identifies the method for modifying the gate.

Our minimization algorithm finds the optimum subset of gates, which are modified. The minimization algorithm

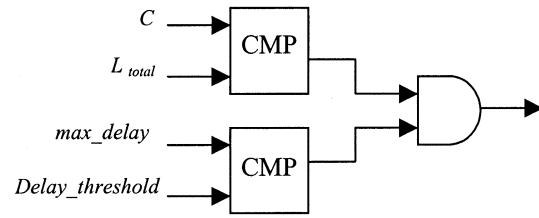


Fig. 11. Considering the delay constraint in leakage minimization.

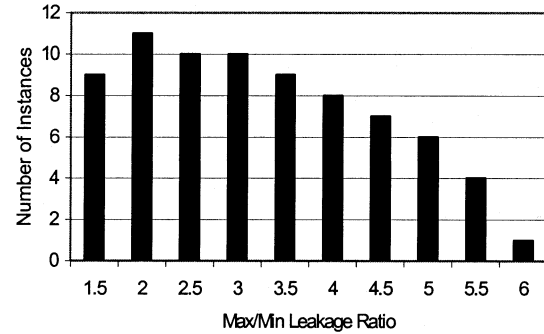


Fig. 12. Distribution of maximum over minimum leakage current.

considers the advantages of modifying the gates in the circuit (which are controlling internal signal as well as reducing the gate leakage due to the stack effect) and weighs them against the disadvantage of additional delay overhead due to the added transistors. In most cases, modifying the gates as in Fig. 6 results in leakage reduction due to the stack effect. Notice that there may exist certain input combinations that result in higher leakage current after modifying the gates in this way. However, the SAT based solution discards those circuit changes that result in higher leakage. More precisely, the leakage of a modified gate is calculated based on its applied input combination. Consequently, the SAT solver accepts a gate modification only if the total circuit leakage (accounting for both leakage of the gate itself and leakage of the fanout gates that this gate is driving) is reduced while satisfying a delay constraint.

V. EXPERIMENTAL RESULTS FOR COMBINATIONAL CIRCUITS

We applied the proposed mechanisms to reduce the leakage currents of the circuits in the MCNC91 benchmark. Each of the circuits was optimized by the SIS *script.rugged* and mapped to a technology library using the SIS mapper. We used an industrial library built in 0.18- μm CMOS technology with a low threshold voltage of 0.2 V and a supply voltage level of 1.5 V. We used HSPICE simulation to report the leakage current of the gates in the ASIC library for all possible combinations of their inputs. We, therefore, started with a full circuit-level characterization of leakage current of all gates. For each benchmark, we obtained the minimum and the maximum leakage currents and their corresponding input vectors using the method described in Section III. Fig. 12 shows the distribution of the ratio of maximum to minimum for all circuits.

Fig. 12 depicts our experimental results where we show the max/min leakage distribution for the MCNC91 benchmark suite. The figure, for example, states that 9 of the benchmarks had a max/min leakage ratio between 1.25 (inclusive) and 1.75

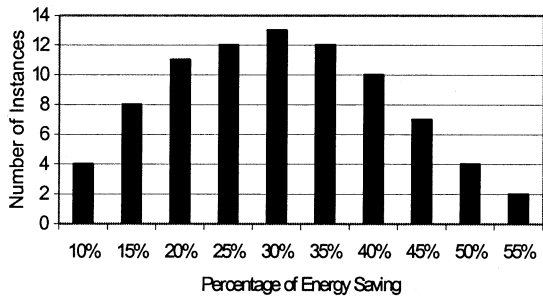


Fig. 13. Energy saving of the input vector control mechanism.

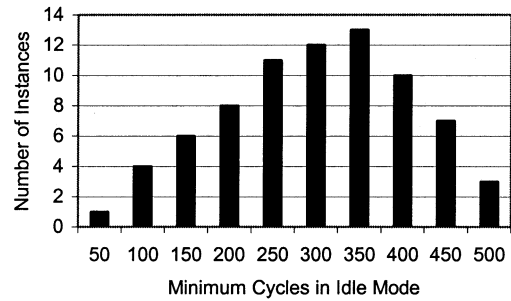


Fig. 15. Minimum number of clock cycles that the circuit should stay in the standby mode for the dynamic leakage control to become effective.

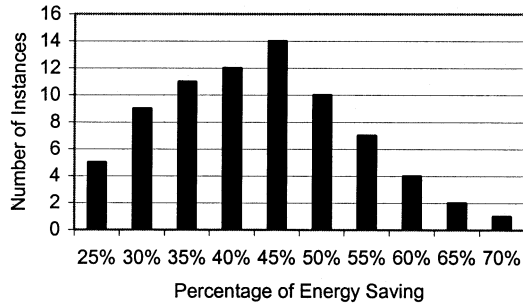


Fig. 14. Energy saving for control point addition mechanism.

(exclusive) whereas 11 had a ratio between 1.75 and 2.25. As it can be seen, the max/min leakage ratio is as high as 6 for some circuits. The degree of leakage saving using this method depends on the degree of controllability of internal nodes via primary inputs. Generally for circuits with more numbers of primary inputs and less numbers of internal nodes the degree of controllability of the circuit leakage is higher and more leakage reduction is possible by controlling primary input in comparison to circuits with less number of primary inputs and more number of internal nodes.

Therefore, driving the circuit that is placed in the idle mode with a random input vector may result in a significant waste of energy compared to riving the circuit with the MLV. Fig. 13 shows the distribution of energy saving achieved by using the input vector control mechanism of Section III.

Fig. 14 shows the distribution of energy saving achieved by using the control point addition mechanism of Section IV-A. As one can see, adding control points to the circuits helps to further reduce the leakage currents.

Switching the inputs of a circuit to its MLV and vise versa consumes some dynamic power. The amount of power saved using our runtime leakage control mechanisms depends on the duration of the standby mode for the circuit. For short standby periods, it is not worthwhile to switch between the current input and the MLV. For long standby periods, the energy savings can become quite significant. To make this statement more precise, we calculated the minimum duration of the idle time above by which power savings becomes possible when “shifting” the MLV in Fig. 15 shows the distribution of this minimum time (in terms of the number of clock cycles in 100 Mhz) for MCNC91 benchmark circuits.

The runtime of the algorithm *LIN_SEARCH_FOR_MLV* depends on the number of quantization levels of leakage values. Obviously more quantization levels result in better accuracy

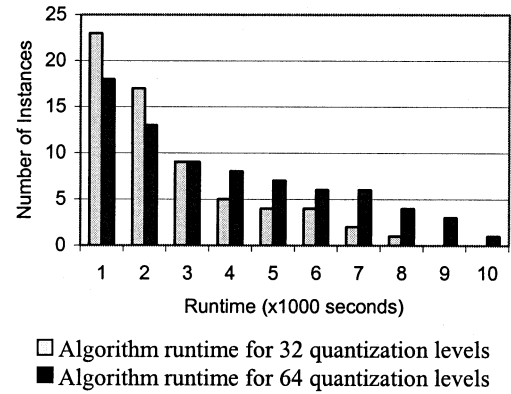


Fig. 16. Algorithm runtime for two different quantization levels.

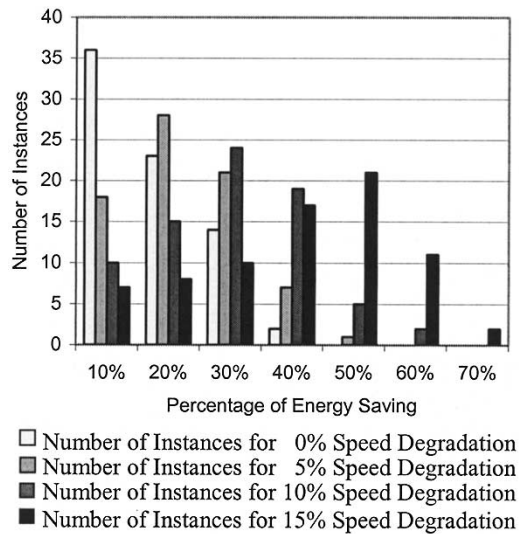


Fig. 17. Distribution of energy savings achieved by using the control point addition mechanism of Section IV-B (modifying gates) under different delay constraints.

and more runtime. Fig. 16 shows the distribution of the runtime of the algorithm for 32 and 64 quantization levels for leakage values (the range of leakage between upper bound and lower bound is quantized to 32 or 64 levels.)

Fig. 17 shows the distribution of energy savings for the MCNC91 suite that is achieved by using the control point addition mechanism of Section IV-B under different delay constraints. When we do not allow any speed degradation, only a small number of gates are changed. As a result, the amount of energy savings on average, is less than 20% for

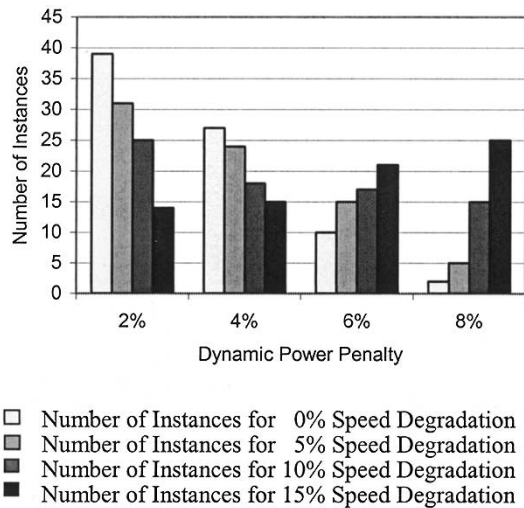


Fig. 18. Dynamic power penalty for the method of Section IV-B under different delay constraints.

all the benchmarks. Increasing the limit on speed degradation helps improve the results as is evident from the figure. For example, with a 15% tolerance on delay, the average energy savings for all the benchmarks is 45–50%. The area overhead is proportional to the number of added transistors and is 15% at most.

We also measured the dynamic power penalty due to the overhead of additional transistors to the circuit, which increases the switching power because of additional capacitance. Fig. 18 shows the dynamic power penalty for “adding control point” mechanism under different delay constraints. As can be seen, when we do not allow any speed degradation, only a small number of gates are changed so the additional capacitance overhead is small and the dynamic power penalty is on average 3%. When more speed degradation is allowed, dynamic power penalty is increased because more control transistors are employed. The dynamic power penalty is tolerable if the leakage saving in the idle mode is significant enough which would be the case if the aggregate idle times are sufficiently larger than the aggregate active times.

In Sections II–IV we discussed some techniques for leakage reduction of combinational circuits. In the rest of the paper we provide some techniques for using the scan chains, which are built in the all VLSI circuits for test purposes to apply minimum leakage vector to the circuit in idle time. Using scan structures gives us the advantage of less hardware overhead and allows us to place the extra hardware in noncritical paths, which result in negligible performance penalty.

VI. SCAN-BASED TESTING

In Fig. 19, we consider a sequential circuit comprised of a combinational circuit and a set of flip-flops.

In the scan-based designs [21], [22], the flip-flops are connected in such a way that they enable two modes of operation: **normal** mode and **test** mode. In the normal mode, the flip-flops are connected as shown in Fig. 19. At each clock cycle, the next state is stored in the flip-flops. In the test mode, the flip-flops are reconfigured and form one or more shift registers, called scan

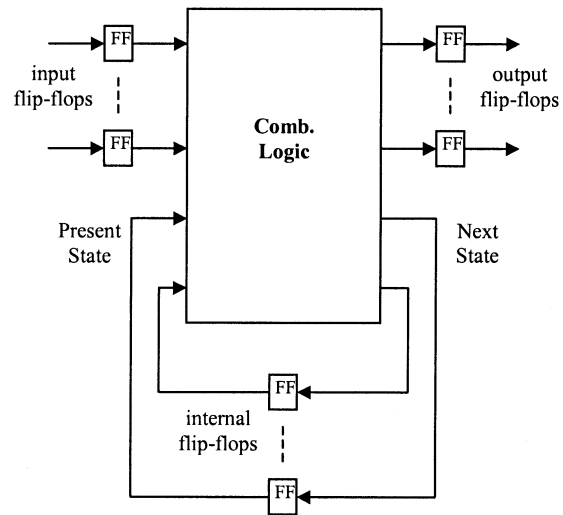


Fig. 19. A general model of a sequential circuit.

registers or scan chains. At each clock cycle the values of the flip-flops are shifted. The values can be observed through the output of the last flip-flop of the scan chain. Furthermore, the values can be shifted into the scan-chain through the input of the first flip-flop in the chain.

In this paper, we assume that all internal and external (input and output) flip-flops are included in the scan chain. This type of circuit is called full-scan. Full scan chains convert the problem of testing a sequential circuit to that of a combinational one. In other words, the input and internal flip-flops can be treated as primary inputs of the circuit, whereas the output and internal flip-flops are considered as the primary outputs. In order to test a circuit, the circuit is first switched to the test mode and the present state value is shifted into the flip-flops. After that, the circuit is switched to the normal mode and operates for one or more cycles under the externally provided input values. In the next step, the circuit is switched back to the test mode and the next state value is shifted out.

As mentioned before, the scan-based test methodology requires the modification of the circuit and addition of a test mode in which the flip-flops are configured as one or more scan chains. For this reason, the flip-flop design must be modified. One way to add the new functionality to the flip-flops is through the addition of a multiplexer with inputs D and D_S , as shown in Fig. 20.

The control input of the multiplexer is controlled by the test signal. This design is referred to as a **multiplexed-input scan flip-flop**. Each flip-flop in the circuit may be replaced by such a flip-flop where its D input is connected to the corresponding state output in the circuit and its D_S input is connected to the output of another flip-flop, which is designated as the predecessor of the current flip-flop in the scan chain. Input D_S of the first flip-flop in a chain is the scan chain input and is denoted by *ScanIn*, while the output of the last flip-flop in the chain is the output of the scan chain and is denoted by *ScanOut*. The input and the output of a chain are connected to an input and an output pin of the chip, respectively. Fig. 21 shows details of a scan chain design. In the figure, the flip-flops are configured as a single chain.

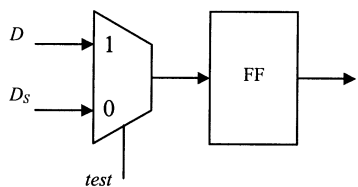


Fig. 20. A multiplexed-input scan flip-flop.

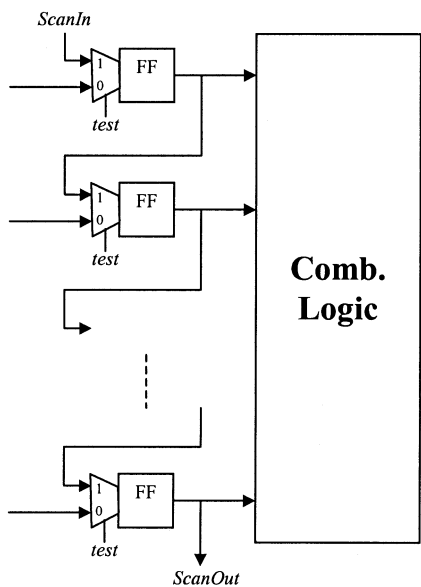


Fig. 21. A generic scan chain structure.

The use of scan allows the desired value to be shifted into each flip-flop, or **scanned in**, using the test mode and scan chains. Hence, the present state of the sequential circuit can be directly controlled. This increases the controllability. After applying a test vector, the values at the state outputs are captured into the flip-flops by configuring them in their normal mode. The captured values are shifted out or **scanned out**, using the test mode and observed at the corresponding scan output pin, *ScanOut*. This means the next state of the sequential circuit becomes observable. This increases observability.

Assuming the flip-flops are configured as a single chain, the following steps are used to apply a test vector.

- 1) The circuit is set into test mode by setting $test = 0$.
- 2) Shift the test vector into flip-flops via *ScanIn* pin by applying $m + k$ clocks, where m and k are the number of input and internal flip-flops, respectively. This causes the test vector be applied to the primary inputs (including present state) of the circuit.
- 3) The circuit is configured in its normal mode by setting $test = 1$ and one clock is applied. This causes the response at the primary outputs (including next state) of the circuit to be captured in the corresponding flip-flops.
- 4) The state response captured in the scan flip-flops is scanned out and observed at the *ScanOut* pin by setting $test = 0$ and applying $k + n$ clocks, where n is the number of output flip-flops.

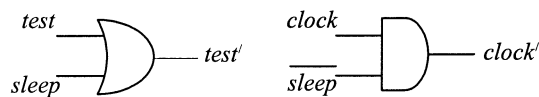


Fig. 22. New test and clock signals.

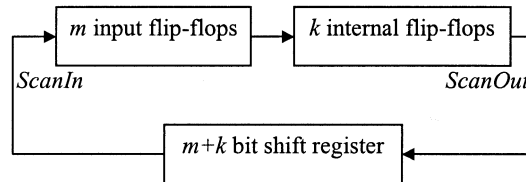


Fig. 23. Configuration of the scan chain in the sleep mode.

VII. USING THE SCAN CHAIN FOR LEAKAGE REDUCTION

In this section we describe how scan chains can be modified to allow us to apply the *MLV* to a sequential circuit when it is in the sleep mode. Because scan-chains provide an easy way to control the values of flip-flops, they can be used to drive the standby circuit with the *MLV*.

A simple way is to shift in the *MLV*, from a memory ($m + k$ bit shift register) into the first $m + k$ flip-flops via the *ScanIn* pin by setting the circuit into the test mode and applying $m + k$ clocks. For this reason the sleep signal, generated by the power management unit, is combined with the test signal to construct the new control input of the multiplexed flip-flops. After shifting in the *MLV*, the clock signal can be disabled to avoid power dissipation in the flip-flops as depicted in Fig. 22.

With such a method, the previous state of the circuit is overwritten by the *MLV*. If the next state or output of the circuit, while switching back to the active mode, is a function of the previous state, then this method will obviously change the functionality of the circuit.

There are many cases in which it is not necessary to know the previous state of the machine upon re-entering the active mode of operation. As an example, consider the floating-point unit of a microprocessor. After executing a floating-point instruction, the unit can be switched back to the idle mode if there are no more floating-point instructions. Upon encountering a floating-point instruction, the unit can be switched back to the active mode. In this case it is not necessary to know the previous state of the unit and the circuit will function properly. On the other hand, there are cases where it is necessary to save the state of the circuit and restore it upon switching back to the active mode. To address this requirement, we propose to add a circuit loop comprised of the input and internal flip-flops and a ($m + k$)-bit shift register as depicted in Fig. 23.

In this way, the state of the circuit can be saved by shifting out the values of the flip-flops via the output of the ($m + k$)th flip-flop (i.e., the last internal flip-flop) in the chain, which can be considered as a *ScanOut* pin, to memory. This memory can be the same ($m + k$)-bit shift register that is used for storing the *MLV*. Shifting in the state can be done the same time the *MLV* is shifted out. Before switching back to the active mode, we need to shift in the previous state, which is saved in the memory, to the

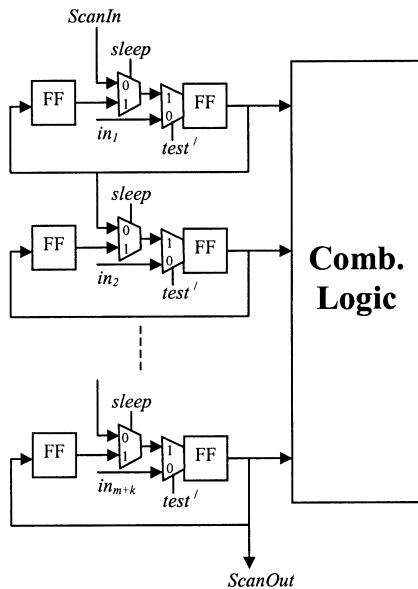


Fig. 24. Modified scan chain for applying MLV in one cycle.

internal flip-flops via the *ScanIn* pin by applying $m + k$ clocks. Simultaneously, the *MLV* captured in the flip-flops of the circuit is shifted into the memory to be used in the next sleep period.

The performance penalty associated with this method is $m + k$ clock cycles, if the length of the sleep period, t , is larger than $m + k$ clock cycles (because it takes $m + k$ clock cycles to load the saved state from the shift register into the flip-flop;) otherwise the performance penalty is $2(m + k) - t$ clock cycles (because we need to return the state values to the flip-flops via the loop.) If we use separate memories ($m + k$ bit shift register for the *MLV* and k bit shift register for the state values,) the performance penalty can be reduced to k clock cycles. If the sleep period is more than $m + k$ clock cycles; otherwise, the performance penalty is $(m + 2k) - t$ clock cycles due to similar reasons.

This method takes advantage of the built in scan structures in the circuit and does not require any modification to the circuit. Therefore, *there is no delay penalty while the circuit is in the active mode*. The fact that this method does not require any changes in the gates of the circuit or any process technology modification makes it very easy to use. On the other hand, it takes several clock cycles to switch between the active and the sleep modes.

Now we describe some modification to the scan chain in order to apply the *MLV* to the circuit in one cycle. For this reason $m + k$ new multiplexers are inserted in the scan chain in such a way that each output of a flip-flop in the scan chain is multiplexed with the corresponding minimum leakage value and the output of the multiplexer is connected to the D_S input of the next multiplexed-input flip-flop as depicted in Fig. 24. The test signal needs to be set to *one* whenever the circuit enters the sleep mode, which can be done by using the circuit in Fig. 22. The added multiplexers can be simplified since one of their inputs is always the minimum leakage value, which is a constant number. This method overwrites the previous state of the circuit with the *MLV*. To solve this problem we add $m + k$ flip-flops and multiplexers controlled by the sleep signal to the circuit, which are

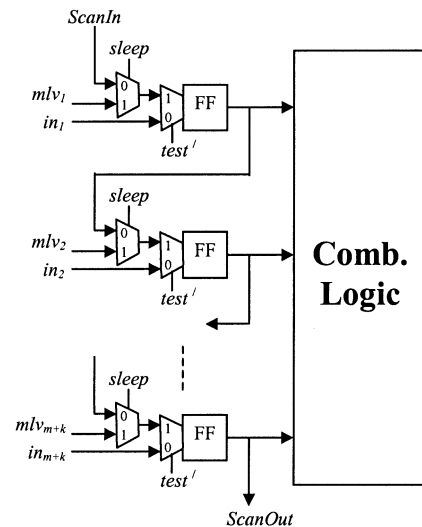


Fig. 25. Adding extra flip-flops for state recovery.

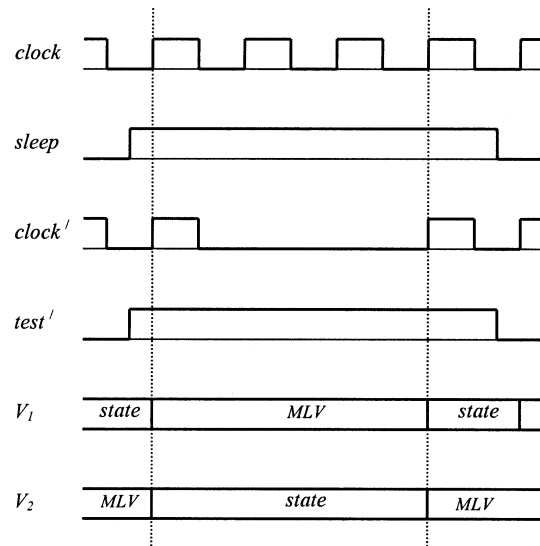


Fig. 26. Timing diagram of control signals.

used to save the *MLV* in the active mode and the previous state in the sleep mode. For this reason we construct a local loop corresponding to each input as shown in Fig. 25.

Disabling the clock as shown in Fig. 22 may not lead to correct results. For correct functionality, the clock needs to be disabled one cycle after entering the sleep mode and it needs to be enabled one cycle before entering the active mode. Fig. 26 shows the appropriate timing of the circuit. In this timing diagram V_1 shows the values captured in the multiplexed-input flip-flops in the scan chain and V_2 shows the values captured in the additional flip-flops. It can be seen that when the sleep signal is high, the current state will be saved in the added flip-flops. At the same time the *MLV* is loaded into the multiplexed-input flip-flops driving the inputs of the combinational circuit. Furthermore, before switching to the active mode, the previous state is captured in the multiplexed-input flip-flops and the *MLV* is concurrently captured in the additional flip-flops.

In some sequential circuits single-latch design is used rather than flip-flop design in which a pair of latches in a master-slave

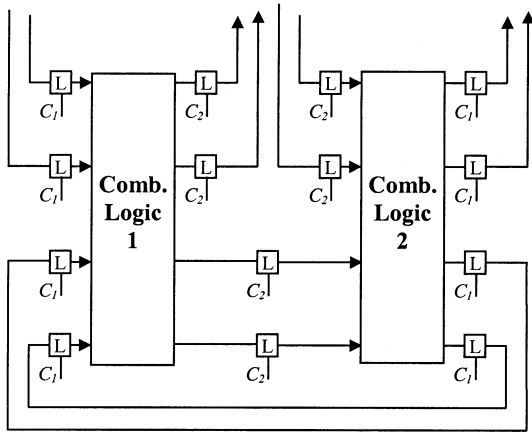


Fig. 27. A single latch sequential circuit.

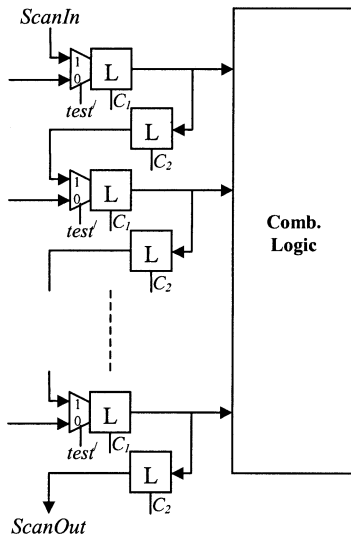


Fig. 28. Scan chain structure for single-latch sequential circuits.

configuration are used. Fig. 27 illustrates the single-latch design in which two nonoverlapping clocks C_1 and C_2 must be used. In such a design if there exists a combinational path from the output of a latch clocked with C_1 to the input of another latch, then that latch must be clocked by C_2 .

Now we describe scan chain design for single-latch circuits. A memory element in a scan design must be capable of selecting the value from one of its two inputs, namely, the state output in the active mode and the scan output of the previous element in the chain in the test mode. Furthermore, since multiple scan elements must be connected as a shift-register, each scan element must have a functionality that is equivalent to that of a flip-flop or a master-slave latch configuration. For this reason each latch is replaced by a multiplexed input latch, similar to the previously described multiplexed input flip-flop. Furthermore, for each latch, an additional latch clocked by a different phase is added to construct the master-slave configuration in the scan chain as illustrated in Fig. 28. In the active mode extra latches hold the *MLV* and the C_3 clock is kept low. When entering the sleep mode by applying a pulse to C_2 , the state is saved in L' latches.

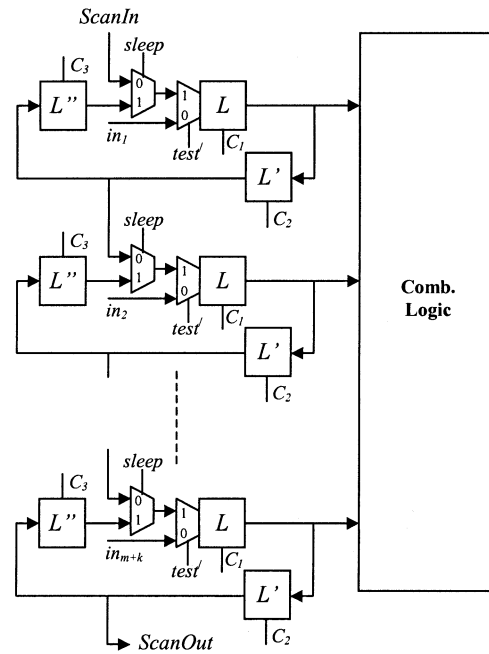


Fig. 29. Adding extra latches and multiplexers for state recovery.

Similar to the previous case, in order to apply the *MLV* in the sleep mode and recover the state when entering the active mode, for each latch, an extra latch clocked by a different clock C_3 and a multiplexer controlled by the sleep signal are added. The extra multiplexers are controlled by the sleep signal as shown in Fig. 29.

Then, by applying a pulse to C_1 and setting $sleep = 1$, which results in $test' = 1$ as shown in Fig. 22, the *MLV* is loaded to L latches driving the combinational circuit. In the next step, applying a pulse to C_3 captures the state values, saved in L' latches, into the L'' latches. This way the data in L and L' latches are swapped via L' latches by applying appropriate pulses to C_1 , C_2 and C_3 . Hence, during the sleep period L'' latches keep the previous state of the circuit. While entering the active mode, the state can be recovered in L latches by swapping data in L'' and L latches by taking a similar approach. Fig. 30 shows the timing diagram of the circuit.

An alternative structure to apply the *MLV* in the sleep mode and recover the state when entering the active mode without extra latches is presented in Fig. 31. In fact instead of extra latches, as in Fig. 29 we need to use extra multiplexers, which correspond to less area and power overhead (Recall that each multiplexer can be implemented by a simple gate as shown in Fig. 4).

Based on this structure in the normal or active mode the inputs are applied to the circuit. In the sleep mode, by first applying a pulse to C_2 the state is stored at L' latches and then by setting $sleep$ signal to one a wake up signal to zero and applying a pulse to C_1 the *mlv* is stored at L latches. When switching back to the operational mode first by setting $sleep$ and $wakeup$ signals to one an applying a pulse to C_1 the previous state is retrieved to L latches via the loop. (The sleep signal needs to be zero only in the test mode.)

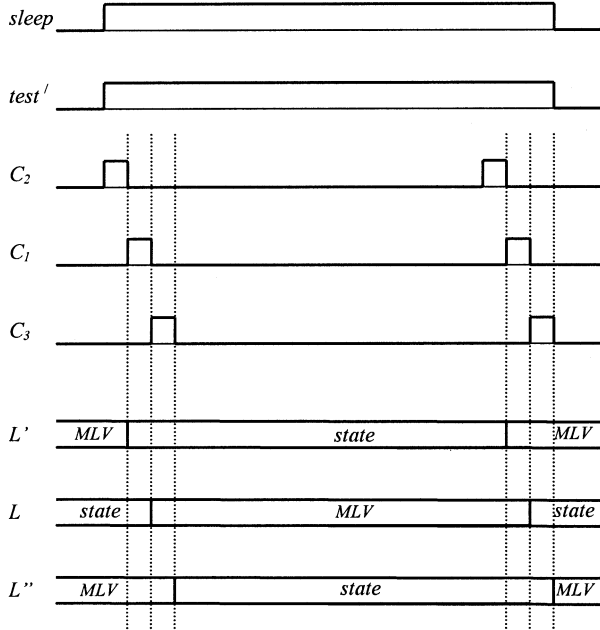


Fig. 30. Timing diagram of control and clock signals.

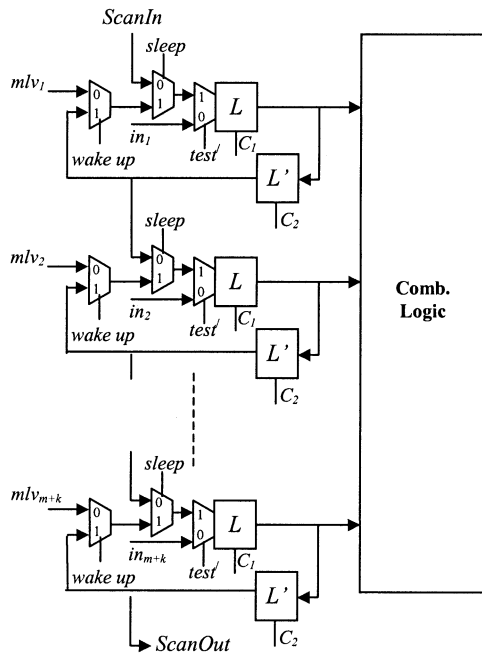


Fig. 31. Input vector control without extra latches.

VIII. EXPERIMENTAL RESULTS FOR SEQUENTIAL CIRCUITS

We applied our leakage reduction methods on **ISCAS89** benchmark circuits. Each method is associated with some delay overhead. We have compared the delay overhead of our methods with the previous method, which does not modify the scan chain of circuits. Table II shows the leakage reduction percentage and minimum cycles in the idle mode required for the method in Fig. 25 to be effective, using input vector control.

The techniques illustrated in Figs. 23 and 24 do not modify the critical paths of the circuit, therefore there is no delay overhead associated with these methods in the active mode. However the method in Fig. 23 is associated with a performance

TABLE II
LEAKAGE REDUCTION PERCENTAGE USING INPUT VECTOR CONTROL

Circuit	Leakage reduction	Min Cycles	Circuit	Leakage reduction	Min cycles
S1196	26%	510	S35932	16%	880
S1238	25%	530	S382	34%	240
S1423	19%	710	S386	27%	490
S1488	31%	330	S400	34%	230
S1494	32%	290	S510	29%	360
S208	36%	190	S5378	19%	690
S27	39%	120	S641	23%	620
S298	35%	220	S713	31%	320
S344	33%	250	S820	33%	240
S349	31%	310	S838	33%	360

penalty and the method in Fig. 24 is not able to recover the state. The method in Fig. 25 is associated with an area overhead and slight delay overhead because of additional capacitive load of extra flip-flops driven by multiplexed-input flip-flops. Table III shows the comparison of delay overhead of our method with standard input control method (using multiplexers in the primary inputs of the combinational circuit, which is on the critical path.). It also includes the dynamic power penalty due to additional hardware. The dynamic power penalty is tolerable if the leakage saving in the idle mode is significant enough which would be the case if the aggregate idle times are sufficiently larger than the aggregate active times.

IX. CONCLUSION

In the first part of this paper, we introduced several methods to decrease the leakage current of a circuit. Our methods do not require any modifications in process technology. Hence, they can be easily used. Furthermore, we presented techniques for reducing the leakage current of a sequential circuit using its minimum leakage vector. Experimental results show, when using our proposed technique, up to 70% savings in the leakage current of combinational circuits can be achieved at the expense of up to 15% delay penalty.

In the second part of this paper, we showed how to modify the scan chain of the circuit and use it to drive the circuit with the minimum leakage vector while the circuit is in standby mode. This effectively eliminates the delay overhead associated with the vector-based methods. Our method results in the loss of the previous state of the sequential circuit. In order to save the state information and restore it upon switching back to the active mode, some extra latches can be added to the circuit. We presented several latch architectures to achieve this goal. Experimental results show, when using our proposed technique, up to

TABLE III
COMPARISON OF DELAY OVERHEAD OF THE PROPOSED METHOD WITH STANDARD METHOD

Circuit	Delay Overhead		Dynamic Power Overhead (our method)
	Standard method	Our method	
S1196	10%	1%	3%
S1238	9%	1%	3%
S1423	4%	1%	4%
S1488	12%	1%	3%
S1494	11%	1%	2%
S208	15%	1.4%	3%
S27	17%	1.5%	2%
S298	13%	1.2%	3%
S344	12%	1%	1%
S349	13%	1.1%	2%
S35932	8%	1%	6%
S382	14%	1.2%	2%
S386	15%	1.2%	3%
S400	13%	1.1%	2%
S510	12%	1%	3%
S5378	11%	1%	5%
S641	10%	1%	4%
S713	9%	1%	3%
S820	12%	1%	2%
S838	13%	1.1%	3%

39% savings in the leakage current of sequential circuits can be achieved at the expense of less than 2% delay penalty.

We did not insert any multiplexer inside the combinational part of the sequential circuit. We only use the scan chain on noncritical paths in order to avoid performance penalty. This is orthogonal to the previous technique used for combinational circuit. Indeed a combination of the two techniques can be used. However, we have opted to focus on the scan chain technique for the sequential circuits since the application of the other technique to the combinational part of the sequential circuits is straightforward.

Note that the leakage reduction may not be very significant for the cases that there is no performance penalty. However, for other cases where some degree of performance degradation is allowed, the leakage saving can be much higher. In addition, the percentage of the leakage current saving that is obtained by the proposed MLV technique increases as the leakage current rises with each new technology node. Intuitively, this phenomenon is due to the fact that higher leakage current in future technology nodes results in an increase of the ratio of the maximum to the minimum leakage in a circuit, which in turn leads to higher leakage current saving when applying the MLV technique.

REFERENCES

[1] S. Borkar, "Design challenges of technology scaling," *IEEE MICRO*, pp. 23–29, July–Aug. 1999.
 [2] A. Abdollahi, F. Fallah, and M. Pedram, "Runtime mechanisms for leakage current reduction in CMOS VLSI circuits," in *Proc. Symp. on Low Power Electronics and Design*, Aug. 2002, pp. 213–218.

[3] —, "Leakage current reduction in sequential circuits by modifying the scan chains," in *Proc. Int. Symp. on Quality of Electronic Designs*, Mar. 2003, pp. 49–54.
 [4] J. Whittimore, J. Kim, and K. A. Sakallah, "SATIRE: A new incremental satisfiability engine," in *Proc. 38th Design Automation Conf.*, June 2001, pp. 542–545.
 [5] A. Ferre and J. Figueras, "Characterization of leakage power in CMOS technologies," in *Proc. IEEE Int. Conf. on Electronics, Circuits and Systems*, vol. 2, 1998, pp. 85–188.
 [6] Z. Cheng, M. Johnson, L. Wei, and K. Roy, "Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks," in *Int. Symp. on Low Power Electronics and Design*, Aug 1998, pp. 239–244.
 [7] D. Duarte, Y. Tsai, N. Vijaykrishnan, and M. J. Irwin, "Evaluating run-time techniques for leakage power reduction," presented at the 7th ASPDAC/15th Int. Conf. on VLSI Design, Jan. 2002.
 [8] M. Johnson, D. Somasekhar, and K. Roy, "Models and algorithms for bounds in CMOS circuits," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 714–725, June 1999.
 [9] Y. Ye, S. Borkar, and V. De, "A new technique for standby leakage reduction in high-performance circuits," in *Proc. Symp. on VLSI Circuits*, 1998, pp. 40–41.
 [10] S. Bobba and I. Hajj, "Maximum leakage power estimation for CMOS circuits," in *Proc. IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999, pp. 116–124.
 [11] M. Johnson, D. Somasekhar, and K. Roy, "Leakage control with efficient use of transistor stacks in single threshold CMOS," in *Proc. 36th Design Automation Conf.*, June 1999, pp. 442–445.
 [12] J. Halter and F. Najm, "A gate-level leakage power reduction method for ultra low power cmos circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1997, pp. 475–478.
 [13] F. A. Aloul, S. Hassoun, K. A. Sakallah, and D. Blaauw, "Robust SAT-based search algorithm for leakage power reduction," in *PATMOS*, Sevilla, Spain, Sept. 2002, pp. 167–177.
 [14] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De, "Design and optimization of low voltage high performance dual threshold CMOS circuits," in *Proc. 35th Design Automation Conf.*, Jun. 1998, pp. 489–494.
 [15] F. Assaderaghi, D. Sinitsky, S. A. Parke, J. Bokor, P. K. Ko, and C. Hu, "Dynamic threshold-voltage MOSFET (DTMOS) for ultra-low voltage VLSI," *IEEE Trans. Electron Devices*, vol. 44, pp. 414–422, Mar. 1997.
 [16] T. Kuroda *et al.*, "A 0.9 V 150 MHz 10 mW 4 mm 2-D discrete cosine transform core processor with variable threshold voltage (VT) scheme," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1770–1779, Nov. 1996.
 [17] A. Chandrakasan, W. Bowhill, and F. Fox, *Design of High Performance Microprocessor Circuits*. Piscataway, NJ: IEEE Press, 2000.
 [18] S. Mutoh, T. Douskei, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V power supply high-speed digital circuit technology with multi-threshold voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, pp. 847–854, Aug. 1995.
 [19] T. Larrabee, "Test pattern generation using boolean satisfiability," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 4–15, Jan. 1992.
 [20] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik, "Efficient conflict driven learning in a boolean satisfiability solver," in *Proc. Int. Conf. on Computer Aided Design*, Nov. 2001, pp. 279–285.
 [21] S. Gupta and N. K. Jha, *Digital System Testing*. New York: Cambridge Univ. Press, 2003.
 [22] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Designs*. New York: Computer Science Press, 1995.

Afshin Abdollahi received the B.S. and M.S. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1997 and 1997. Currently heworking toward the Ph.D. degree at Electrical Engineering Department at University of Southern California, Los Angeles.

His main research interests are low power logic synthesis and leakage current reduction in VLSI circuits.

Farzan Fallah (S'97–M'99) received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1992 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1996 and 1999, respectively.

He joined Fujitsu Laboratories of America, Inc., Sunnyvale, CA, in April 1999 as a Researcher. He has authored or coauthored several papers on design verification and validation. His current research interests are in the area of computer-aided design of integrated circuits with emphasis on logic synthesis, design verification, and validation.

Dr. Fallah is a member of the ACM. He received the Best Paper Award at the Design Automation Conference in 1998.

Massoud Pedram (S'88–M'90–SM'98–F'01) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, in 1986, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989 and 1991, respectively.

He then joined the Department of Electrical Engineering—Systems at the University of Southern California where he is currently a professor. His current work focuses on developing computer aided design methodologies and techniques for low power design, synthesis, and physical design.

Dr. Pedram has served on the technical program committee of a number of conferences, including the Design automation Conference (DAC), Design and Test in Europe Conference (DATE), Asia-Pacific Design automation Conference (ASP-DAC), and International Conference on Computer Aided Design (ICCAD). He served as the Technical Co-chair and General Co-chair of the International Symposium on Low Power Electronics and Design (SLPED) in 1996 and 1997, respectively. He was the Technical Program Chair and the General Chair of the 2002 and 2003 International Symposium on Physical Design. He has published four books, 60 journal papers, and more than 150 conference papers. His research has received a number of awards including two ICCD Best Paper Awards, a Distinguished Citation from ICCAD, a DAC Best Paper Award, and an IEEE TRANSACTIONS ON VLSI SYSTEMS Best Paper Award. He is a recipient of the NSF's Young Investigator Award (1994) and the Presidential Faculty Fellows Award (a.k.a. PECASE Award) (1996). He is a Member of the Board of Governors for the IEEE Circuits and systems Society, an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, and the IEEE Circuits and Systems Society Distinguished Lecturer Program Chair. He is also an Advisory Board Member of the ACM Interest Group on Design Automation, and an Associate Editor of the *ACM Transactions on Design Automation of Electronic Systems*. For more information, please go to URL address: <http://atrk.usc.edu/~massoud/>.