Leakage Control With Efficient Use of Transistor Stacks in Single Threshold CMOS

Mark C. Johnson
Rose-Hulman Institute of Technology
5500 Wabash Avenue
Terre Haute, IN 47803-3999, USA
Mark.Johnson@Rose-Hulman.Edu

Dinesh Somasekhar
Purdue University
1285 EE Building
West Lafayette, IN 47907-1285, USA
somasekh@ecn.purdue.edu

Kaushik Roy
Purdue University
1285 EE Building
West Lafayette, IN 47907-1285, USA
kaushik@ecn.purdue.edu

ABSTRACT

The state dependence of leakage can be exploited to obtain modest leakage savings in CMOS circuits. However, one can modify circuits considering state dependence and achieve larger savings. We identify a low leakage state and insert leakage control transistors only where needed. Leakage levels are on the order of 35% to 90% lower than those obtained by state dependence alone.

1. INTRODUCTION

In response to the growing need for low voltage, high performance, low leakage systems, a few circuit level approaches to leakage control have already been developed. Some methods depend on the use of multiple threshold voltages. Low threshold voltage transistors are used where needed to improve performance. High threshold devices are then used for leakage control. The MTCMOS technique [10], [9] isolates low threshold voltage circuits from the power and ground rails using high threshold voltage devices. There is a performance penalty since the high threshold transistors appear in series with all switching current paths in the circuit. Another approach is to use low threshold devices in the critical path of a circuit and high threshold voltage devices elsewhere to achieve leakage savings without a performance penalty [12]. One can also limit leakage through dynamic control of the threshold voltage. Threshold voltage is lowered when a circuit is active and elevated when idle. This can be accomplished by substrate biasing [7], [6] or with dual gate silicon-on-insulator (SOI) technologies [11], [3].

We propose an approach which does not require multiple threshold voltages or substrate biasing schemes, but takes advantage of the natural leakage behavior in stacks of MOS transistors [5], [1] to reduce sleep mode leakage while avoiding active mode performance loss. We first identify a circuit input vector that will put most of the circuit into a low leakage state. In general, the low leakage state occurs when as many MOS transistors as possible are turned off in each leakage path. We then insert transistors for leakage control only in those leakage paths where it was not possible to turn off more than one transistor. Others have exploited state dependence for energy reduction [4] and for leakage control in IDDQ test [8]. However, it has been observed that often the variation of leakage with respect to circuit state alone is 50% or less [1].

2. SELECTION OF MINIMUM LEAKAGE INPUT VECTORS

To identify minimum leakage input vectors, we define leakage "observability" measures which indicate the degree to which the value of a particular circuit input is observable in the magnitude of leakage from the power supply.

2.1 Leakage Observability Measures

Let $Lobs_i(k, \mathbf{w})$ represent the leakage observability of input i to circuit k, given a partially specified input vector $\mathbf{w}$. If input i is already specified in $\mathbf{w}$, then we set $Lobs_i(k, \mathbf{w}) = 0$. For our purposes, observability values are only useful for inputs that have not already been fixed to a particular value. If input i is not already specified in $\mathbf{w}$, then the "leakage observability of input i" is determined as follows.

Let $Lavg_i^v(k, \mathbf{w})$ represent the portion of overall average leakage cost attributable to forcing the value v on input i. If g is a logic gate, we define $Lavg_i^v(g, \mathbf{w})$ to be the average leakage of all possible leakage states of gate g divided by the number of inputs not already specified in $\mathbf{w}$. "All possible leakage states," is restricted to the set of states permitted by the partially specified input vector $\mathbf{w}$ and the assignment of value v to input i. Leakage observability is calculated from average leakage costs in the following manner:

$$Lobs_i(k,\mathbf{w}) = | \text{Lavg}_i^1(k, \mathbf{w}) - \text{Lavg}_i^0(k, \mathbf{w}) | \quad (1)$$

In larger networks of logic gates, it is necessary to define how the average leakage costs for each gate are represented at the primary inputs of the larger network. Let $L'avg_i^v(g, \mathbf{w_g})$ represent the average leakage cost at input i of gate g, including leakage costs associated with logic gates in the fanout tree of g. We will refer to this as the "fanout leakage cost". $\mathbf{w_g}$ represents any inputs to g which are already specified. The fanout leakage costs at the primary outputs of network k are defined to be zero. Consequently, for the last level of gates in the network, $L'avg_i^v(g, \mathbf{w_g}) = Lavg_i^v(g, \mathbf{w_g})$. For all other gates in the network, the fanout leakage costs are recursively defined by equation (2).

$$L'avg_i^v(g, \mathbf{w}_g) = \frac{1}{N_{free}} \underset{\forall x \in I(g, \mathbf{w_g} \& \{i=v\})}{Average} (F(g,\mathbf{x})) \qquad (2)$$

where

$$F(g,\mathbf{x}) = L(g,\mathbf{x}) + \sum_{\forall j \in fanout(g)} L'avg_j^{g(\mathbf{x})}(h_j, \mathbf{w}_{h_j})$$

and **x** is a fully specified input vector to gate *g*. $I(g,\mathbf{w}_g \,\&\, i = v)$ is the set of input vectors compatible with the partially specified input vector $\mathbf{w_g}$ and the assignment of value *v* to input *i*. $L(g,\mathbf{x})$ is the leakage of gate *g* for input vector **x**. *fanout(g)* is the set of inputs to other gates driven by the output of gate g. j represents an input to gate h_j which is connected to an output of gate g. $g(\mathbf{x})$ is the logic value output by gate *g* given input vector **x**. N_{free} is the number of inputs to gate *g* that are not specified in $\mathbf{w}_g$. If input *i* is already specified in $\mathbf{w_g}$, then $L'avg_i^v(k,\mathbf{w}) = 0$.

2.2 Heuristic search for leakage bounds

Identification of a minimum leakage input vector can be shown to be NP-hard by a polynomial time transformation from the 3-CNF circuit satisfiability problem, which is NP-complete [2]. Consequently, we devised a heurstic using the value of $L'obs_i$ at each primary input (PI) to guide selection of input values.

1. Evaluate leakage observability measures
2. Put each PI in a priority queue with |L'obs| as the priority measure
3. Until the priority queue is empty, do...
   3.1 Pop a new PI(*i*) off of the queue.
   3.2 If $L'avg_i^1 > L'avg_i^0$, set PI=0, else set PI=1
   3.3 Update the circuit state for the new input.
   3.4 Update the leakage observability measures.
   3.5 Re-sort the priority queue.
4. Done.

The priority queue is used to make sure that the next PI set is the one for which we have the strongest indication as to the preferred input value. Circuit state and leakage costs are updated and the queue is re-sorted each time after another PI is set because setting a single PI can radically change the average leakage cost for remaining inputs. In practice, we found that one can greatly reduce the frequency of cost function updates without serious loss of effectiveness.

3. STACKING TRANSISTOR INSERTION

Insertion of stacking transistors is a technique which exploits the state of a circuit once a minimum leakage input vector has been applied. For each gate that is in a high leakage state, we insert a leakage control transistor in series between the power supply line and the pull-up network or between the ground line and the pull-down network. Figure 1 illustrates the principle. Circuits (a) and (b) are in a high leakage state, i.e., no leakage path passes through more than one transistor which is turned off. Circuits (c) and (d) illustrate the insertion of a leakage control transistor (which can be shared by multiple gates).

3.1 Leakage control insertion algorithm

To estimate the potential costs and benefits of leakage control transistor insertion, we first use the algorithm described in section 2.2 to select a low leakage input vector. A post-processing phase then identifies logic gates in a high leakage state that can be modified to use a leakage control transistor. The algorithm is structured as follows:

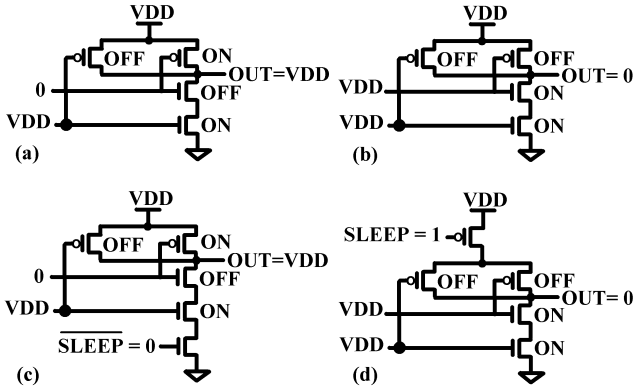

Figure 1. Logic gates with and without leakage control transistors

1. Perform critical path and slack analysis
2. Select minimum leakage inputs (greedy algorithm)
3. For each logic gate in the design, do...
   3.1 If gate not in a high leakage state, skip it.
   3.2 If gate in a critical path (optional), skip it.
   3.3 Otherwise, put the gate in a priority queue where highest *leakage x slack* gets priority
4. Repeat until the priority queue is empty...
   4.1 Pop the highest priority gate from the queue
   4.2 Re-evaluate delay of gate to include leakage control
   4.3 If there is sufficient slack to permit this delay,
      4.3.1 If gate output=1, connect gate to NMOS leakage control
      4.3.2 else if gate output=0, connect to PMOS
      4.3.3 Update slack calculation in fan-in and fan-out trees of the logic gate
      4.3.4 Set leakage of the logic gate to 0 (this leakage will be accounted in step 5)
5. Re-evaluate total leakage of logic network
   5.1 Accumulate sum of individual gate leakages
   5.2 Estimate leakage through control transistors
6. Done.

In step 1 a critical path analysis is performed like that described in [12]. This gives us a timing slack value for each logic gate. Step 2 gives us the state of each logic gate when the minimum leakage vector is applied to the circuit. Step 3 identifies any logic gates that can not or would not be connected to a leakage control transistor. This could be because the gate is already in a low leakage state, or it could be because the gate is in a critical path and cannot tolerate any increase in delay. In step 4, each logic gate is connected, if possible, to common NMOS or PMOS leakage control transistors. We make the conservative

MCNC Benchmark:	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
# gates	39	109	92	136	269	340	405	1898	527	2285
# PI's	25	201	132	192	133	138	199	133	99	257
Min Bound: [µA]	3.1	4.2	3.9	6.3	17.1	16.4	13.4	74.5	23.9	140
Min Vector: [µA]	4.9	9.9	14.2	14.9	36.4	40.5	46.5	296	126	398
HSPICE: [µA]	4.8	9.7	13.3	13.9	31.2	37.7	41.9	266	118	N/A
Leak Control: [µA]	1.6	3.8	5.4	6.7	6.0	15.9	22.6	76.0	18.7	66.1
HSPICE: [µA]	1.6	3.9	5.1	6.0	5.8	13.9	19.6	69.3	18.1	N/A
# gates controlled	19	28	18	38	182	100	97	996	335	1530
NMOS Control: [µm]	7.2	14.4	30.6	7.2	61.2	23.4	19.8	416	160	513
PMOS Control: [µm]	10.8	18	0	46.8	90	130	133	673	425	1260
Logic Cap. [pF]	2.0	8.9	5.7	10.0	16.4	27.1	34.3	137	42.4	131
Sleep Mode Cap. [fF]	39	63	63	96	260	256	255	1800	966	2900
CPU Time: [sec]	1	15	7	17	36	68	113	1352	141	1899

Table 1. Detailed Leakage Control Results – 30% Sizing

MCNC Benchmark:	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
Min Vector: [µA]	4.9	9.9	14.2	14.9	36.4	40.5	46.5	296	126	398
Min Bound: [µA]	3.1	4.2	3.9	6.3	17.1	16.4	13.4	74.5	23.9	140
100% Sizing: [µA]	2.2	4.8	6.8	8.1	11.0	20.2	26.8	113	37.1	123
50% Sizing: [µA]	1.8	4.1	5.9	7.1	7.6	17.3	23.9	87.8	24.6	84.2
30% Sizing: [µA]	1.6	3.8	5.4	6.7	6.0	15.9	22.6	76.0	18.7	66.1
10% Sizing: [µA]	1.5	3.4	4.9	6.2	4.2	14.4	21.1	62.8	12.2	46.0

Table 2. Leakage vs. Control Transistor Sizing

assumption that the delay impact of the shared leakage control transistor is equivalent to effect of a seperate nominally sized transistor inserted in each high leakage path.

Once leakage control has been applied wherever possible, the overall leakage estimate is revised in step 5. For those gates for which leakage control was not applied, the leakage can be accumulated as the sum of leakages for each gate, taken from look-up tables giving the leakage of each gate as a function of the input vector. For those gates that are connected to an NMOS (or PMOS) leakage control transistor, we generate an equivalent circuit to which the transistor stack based leakage model [5] can be applied.

4. RESULTS

For MCNC benchmarks "I1" through "I10", table 1 presents detailed leakage control results for the assumption that leakage control transistors can be sized to 30% of the total transistor width being controlled (the sum of the widths of the transistors which are turned off and are in leakage paths being controlled). "Min Bound" is a lower bound on leakage obtained as the sum of the individual leakages of each logic gate. "Min Vector" is the leakage when a minimum leakage input vector is applied to the circuit. The "Min Vector" results were obtained using the greedy method presented in section 2. For minimum leakage vector selection, gates in the critical path were weighted more heavily (5x) than gates off critical path. Rows labeled "HSPICE" present static leakage measurements made by means of HSPICE simulation. The row labeled "Leakage Control" gives leakage estimates for circuits where leakage control transistors have been used.

"NMOS" and "PMOS Control" give the total size of the leakage control transistors for pull-down paths and pull-up paths respectively. "Logic Capacitance" gives the sum of all junction and gate capacitances that can be charged or discharged during normal active mode operation of the circuit. "Sleep Mode Capacitance" specifies the total capacitance switched when all leakage control transistors are turned off (sleep mode) or on (active mode). This assumes that the minimum leakage vector has already been applied to the circuit before the leakage control transistors switch. "CPU Time" gives the total CPU time for the entire computation of results in each column including initial parsing of input files. The results in this table were generated an a Sun Sparc Ultra-30 workstation.

Table 2 presents a comparison of leakage estimates for different leakage control transistor sizing assumptions. "Min Bound" and

"Min Vector" results from table 1 are included to facilitate comparison. Most values other than leakages presented in table 1 are unaffected by the transistor sizing assumption.

Figure 2 summarizes the leakage results from tables 1 and 2. Each figure resembling an "I" spans the range of leakage values that be reached by choice of input vectors. The wide horizontal bar below the "I" marks the lower bound on leakage for the unmodified circuit (the sum over all logic gates of the gate level minimum leakage). The small cross-hair symbols (resembling a "+") denote leakage for various control transistor sizings. In each column, from top to bottom, the cross-hair symbols correspond to the 100%, 50%, 30%, and 10% transistor sizing assumptions.

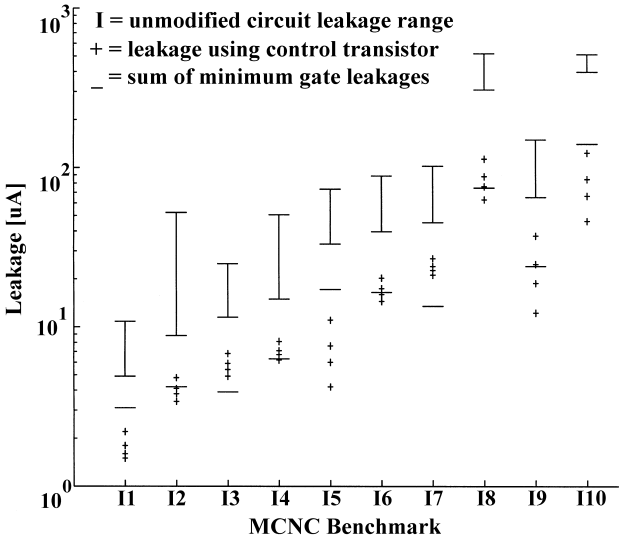

Figure 2. Summary of leakage control results

Some interesting observations can be drawn from these results in The most important is the effect of leakage control transistor insertion on overall leakage. Compared to the leakage obtained when we apply a low leakage input vector, we obtain anywhere from a factor of 2x (circuit I7) to better than 5x (circuit I5) reduction in leakage. Furthermore, the leakage with control transistors was always similar in value or sometimes much lower than the lower bound on leakage for the unmodified circuit. One should also note that the gate capacitance of the sleep mode transistors is much smaller than total logic circuit capacitances. This suggests that the major cost associated with putting a circuit into sleep mode will be the signal path switching associated with application of the minimum leakage input vector rather than with the switching of the leakage control transistors.

4.1 CONCLUSIONS

For low supply voltage, low threshold voltage logic, the use of a minimum leakage vector together with stacking transistor insertion is a promising option for leakage control. It offers a leakage reduction on the order of 35% to as much as 90% relative to an unmodified circuit with a minimum leakage vector applied. This is accomplished with no direct impact on performance since the technique is only applied to gates which are off the critical path. There also should be minimal direct impact on switching power since no capacitances are added to logic signal paths in the circuit.

5. REFERENCES

[1] Chen, Z., Johnson, M., Wei, L., and Roy, K. Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks. Proceedings of the Symposium on Low Power Design and Electronics (1998), 239-244.

[2] Cormen, T.H., Leiserson, G.E., and Rivest, R.L. *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 1990.

[3] Gil, J., Je, M., Lee, J., and Shin, H. A high speed and low power SOI inverter using active body bias. Proceedings of the Symposium on Low Power Electronics and Design.(1998), 59-63.

[4] Halter, J.P., and Najm, F. A gate-level leakage power reduction method for ultra-low-power CMOS circuits. Proceedings of the IEEE Custom Integrated Circuits Conference (1997), 475-478.

[5] Johnson, M.C., Somasekhar, D., and Roy, K. A model for leakage control by MOS transistor stacking. Tech. Rep. TR-ECE 97-12, Purdue University, School of Electrical and Computer Engineering, 1997.

[6] Kobayashi, T., and Sakurai, T. Self-adjusting threshold-voltage scheme (SATS) for low-voltage high-speed operation. Proceedings IEEE Custom Integrated Circuits Conference (1994), 271-274.

[7] Kuroda, T., et al. A 0.9v 150MHz 10 mW *4mm*^{2} 2-D discrete cosine transform core processor with variable-threshold-voltage scheme. Proceedings IEEE International Solid-State Circuits Conference (1996), 166-167.

[8] Maxwell, P.C., and Rearick, J.R. A simulation-based method for estimating defect-free IDDQ. Digest of Papers, IEEE International Workshop on IDDQ Testing (1997), 8O-84.

[9] Mutoh, S., et al. 1-v power supply high-speed digital circuit technology with multithreshold-voltage CMOS. IEEE Journal of Solid-State Circuits, vol.30, no.8 (Aug. 1995), 847-853.

[10] Shigematsu, S., et. al. A 1-V high-speed MTCMOS circuit scheme for power-down applications. IEEE Symposium on VLSI Circuits Digest of Technical Papers (1995), 125-126.

[11] Vieri, C., et al. SOIAS: Dynamically variable threshold SOI with active substrate. Proceedings of the Symposium on Low Power Electronics (1995), 86-87.

[12] Wei, L., Chen, Z., Roy, K., Johnson, M.C., Ye, Y., and De, V. Design and optimization of dual threshold circuits for low voltage low power applications. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.7, no.1 (March 1999), 16-24.