# Simultaneous Vt Selection and Assignment for Leakage Optimization

Ankur Srivastava
Department of ECE, University of Maryland, College Park
ankurs@eng.umd.edu

## Abstract

This paper presents a novel approach for leakage optimization through simultanous $V_t$ selection and assignment. $V_t$ selection implies deciding the right value for $V_t$ and assignment implies deciding which gates should be assigned which thresh-hold value. The proposed algorithm is a general mathematical formulation that can be trivially extended to multiple thresh-hold voltages (more than two). Traditional leakage optimization strategies either assume the prespecification of thresh-hold values or are good only for two thresh-holds. The presented formulation is based on linear programming approach under the piecewise linear approximation of delay/leakage vs thresh-hold curves. The algortihm was incorporated in SIS. Experimental results indicate that on some benchmarks having more that two thresh-holds was beneficial for leakage.

**Categories & Subject Descriptors:** B.6.3 [LOGIC DESIGN] Design Aids-Optimization
**General Terms:** Algorithms, Theory, Performance.
**Keywords:** Leakage Power, Linear Programming, Thresh-hold Voltage

## 1. INTRODUCTION

Continuous shrinking of feature sizes has enabled scaling of voltage and hence massive reduction in dynamic power. This reduction in supply voltage must be accompanied with a proportional reduction in thresh-hold voltage in order to contain adverse effects on delay. This thresh-hold reduction however is accompanied by an exponential increase in leakage current and hence leakage power. Leakage power/current can primarily be attributed to the stored charge in devices that are off which keep conducting (leaking) current. As technology scales, the importance of leakage power/current is becoming more and more significant. In fact it is becoming the dominant component of overall chip power. New optimization methodologies are desired which optimize the leakage directly without effecting delay.


*ISLPED'03,* August 25–27, 2003, Seoul, Korea.


The current state of research has proposed many strategies for reducing leakage. Having multiple thresh-holds on chip is one of the most significant among them. One approach considers having a high thresh-hold transistor in series with the gates so that leakage current could be reduced [4]. Another approach suggests having gates on critical (slow) paths with low thresh-hold and non-critical paths with high-thresh-hold [7], [5]. Most of the existing work assumes that the designer is interested in only two thresh-hold voltages on chip. Having multiple thresh-hold voltages on silicon is more of an economic issue than a technical one. Each extra thresh-hold voltage would require an additional step in fabrication process which costs money. Hence most of the existing research has focused only on two thresh-hold voltages.

This paper presents a novel approach for leakage optimization problem by assigning thresh-hold voltage to each gate in the circuit. Our approach is similar to [5], [7] since we assign thresh-hold voltages to gates. The key disadvantages of existing approaches are as follows. Firstly their optimization strategy is good only for two thresh-holds. Secondly they assume the two thresh-hold voltages to be prespecified. They do not allow the designer to investigate the possibility of having more than two thresh-holds (even though it is more expensive). This paper presents a general mathematical formulation for optimizing leakage current/power. It does not assume any predecided number of thresh-holds or specific thresh-hold voltage values. It optimizes leakage by solving the thresh-hold selection and assignment problem simultaneously. Thresh-hold selection problem tries to select the exact values for thresh-holds. Thresh-hold assignment problem assigns gates to a certain thresh-hold. Given a circuit, a delay constraint and a constraint on number of thresh-holds, the algorithm will automatically select the correct values for thresh-holds and assign gates to these thresh-holds. The designer can easily experiment with more than two thresh-holds. The approach is based on the piecewise linear approximation of delay/leakage vs thresh-hold curves. The formulation is based on linear programming and is hence polynomial in runtime. The formulation is also optimal under the piecewise linear approximation and if number of thresh-hold voltages is not a constraint. The approach of [7] is similar in approach to ours since they too try to select the correct thresh-hold. But their approach is good only for two voltages. Our formulation is a generalized approach for any number of thresh-hold voltages.

Experimental results were conducted in the SIS framework with the MCNC benchmark suite. We experimented

with one, two, four and unlimited thresh-holds. Results showed that for many benchmarks leakage for two and four thresh-holds is very similar. But for a few of the benchmarks (larger ones), leakage of four thresh-hold voltages was significantly better than two.

The rest of the paper is organized as follows. Section 2 discusses the related work, mathematical formulations and basics of leakage. Section 3 contains our complete mathematical and algorithmic formulation for simultaneous thresh-hold selection and assignment. Section 4 contains the experimental results and section 5 contains the conclusion.

# 2. PRELIMINARIES

## 2.1 Related Work in Leakage Optimization

Leakage optimization problem has been a topic of active research over the last few years. This subsection will briefly overview the major results of the existing work. Scaling of supply voltage forces the scaling of thresh-hold voltage $V_t$ inorder to contain the adverse effects on delay. This lowering of thresh-hold has an exponentially detrimental effect on leakage current/power because of an exponential dependence [6]. Leakage current is becoming increasingly important component of overall chip power dissipation. The use of two thresh-hold voltages has been proposed [3] which could be physically implemented by using an additional step in fabrication process. More than two thresh-hold voltages would require more complicated manufacturing technology which might make fabrication more expensive. [4] and many others consider the inclusion of a high thresh-hold sleep transistors which will reduce the leakage in sleep mode. There are significant issues regarding the control of these sleep transistors which have not be addressed. Another approach is to have low thresh-hold voltages on critical paths and high thresh-hold on non-critical paths [7], [5]. Other approaches include considering the state in which a gate exists. The leakage current strongly depends on state of the circuit. Some researches have tried to achieve the "optimal" state configuration of the circuit to minimize the leakage power.

## 2.2 Leakage in Deep-Submicron Revisited

Leakage current of a MOS transistor according to the BSIM model [2] can be approximated as follows

$$I_{leak} = Ae^{q(V_{gs}-V_t)/nkT}(1 - e^{-qV_{ds}/kT}) \quad (1)$$

where $A = \mu o C_{ox}(W_{eff}/L_{eff})(kT/q)^2 e^{1.8}$ and $C_{ox}$ is the gate oxide capacitance per unit area and $V_t$ is the thresh-hold voltage. It can be seen that leakage current is exponentially dependent on thresh-hold. On the other hand, delay of a MOS transistor follows the following approximate relation w.r.t $V_t$ [7], [2], [6]

$$t_d = 2C_{load}V_{dd}/(\beta)(V_{dd} - V_t)^{\alpha}) \quad (2)$$

where $\alpha$ is around 1.3 for short channel and 2 for long channel devices.

Leakage current of a CMOS circuit is simply the sum total of the leakage currents of all gates. The leakage of a CMOS gate depends on the number of transistors that are turned off and hence on the inputs. For example if a NAND gate has both NMOS transistors off (input = 00). Since these transistors are in a stack, the leakage current will be small, whereas if both PMOSs are off (input = 11) then the two off transistors are connected in parallel and the leakage is large.

## 2.3 The Gate-Level Circuit Delay and Leakage: Problem Formulation

Given a gate level circuit and a delay constraint, we now express the leakage optimization problem as a function of gate thresh-hold voltages. We assume that all transistors in the gate have the same thresh-hold voltage. Close observation of the formulation will reveal that this does have to be the case necessarily. This formulation can be trivially generalized to the case presented by [5] where each NMOS, PMOS pair have the same thresh-hold.

Let $d_{i,g}(V_t(g))$ represent the internal delay of gate $g$ from pin $i$ to the output. The term also indicates its dependence on thresh-hold.

$$Minimize \sum_{\forall gates} \quad \sum_{g} I_g(V_t(g)) \quad (3)$$

$$a_g = A_g \qquad \forall \quad g \in PI \quad (4)$$

$$a_g = max_{\forall \quad fanins-of-g}(a_i + d_{i,g}(V_t) | fanin \ \ at \ \ i{-}th \ \ input) \quad (5)$$

$$a_g \leq R_g \qquad \forall g \in PO \quad (6)$$

$$V_t(g) - min \leq V_t(g) \leq V_t(g) - max \qquad \forall g \quad (7)$$

The equations above contain the formal description of the general-leakage optimization problem. Essentially, we would like to assign thresh-hold voltages to each gate such that the required time constraint (equation 6) at all the primary outputs (PO) is satisfied and the overall leakage current/power is minimized. Equation 4 assigns an arrival time to each of the primary input. Equation 5 signifies the propagation of arrival time throughout the circuit. Note that both gate delays and leakage currents are functions of gate thresh-hold.

This formulation assumes that the number of different thresh-hold voltages available on the chip are potentially infinity. Hence each gate can be assigned a separate thresh-hold. Existing research reports that this may not be an economically viable option. Typically, two predecided voltage levels are assumed and gates/transistors are assigned voltages from this set [5]. This is called the dual-leakage optimization problem. There are two fundamental questions that this paper tries to address. Firstly how do we choose which thresh-hold voltages should be present on chip : **the voltage selection problem**. Secondly which gates should have which thresh-hold voltage: **the voltage assignment problem**. We present a general framework for solving this problem simultaneously. Our formulation can be trivially extended to consider more than two thresh-hold voltages if the designer/technology wants to support more than two. The key differences between our and existing approach is as follows: firstly instead of assuming a predefined set of thresh-hold voltages, our formulation automatically selects the appropriate voltage values. Moreover our formulation can be easily generalized to more than two thresh-holds

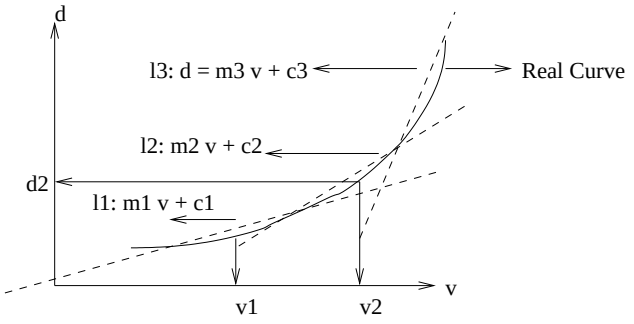


**Figure 1: Piecewise linear approximation**

which is in stark contrast with the approach presented in [7]. The strategy of [7], although picks appropriate thresh-holds, is good only for two voltages.

# 3. SIMULTANEOUS SELECTION & ASSIGNMENT

As mentioned before, we try to solve the following problems simultaneously

1. Thresh-hold Selection: What should be the values of the thresh-hold voltages
2. Thresh-hold Assignment: Which gates should be assigned which thresh-hold voltage

Previous section defined the leakage optimization problem formally. Let us make a few observations on the presented formulation. Firstly, both delay and leakage have a convex dependence on thresh-hold voltage (equation 1, 2). Hence the leakage optimization problem entails a convex set of constraints along with a convex objective function. (Note: For brevity, we have omitted the detailed discussion on how the present formulation is convex.). The generalized leakage problem can be solved optimally using any convex optimization tool. Of course the number of distinct thresh-hold voltages may become too many, hence making the solution impractical. But this solution does give a lower bound on the leakage optimization problem. Although convex programming techniques are very fast, they may not be efficient for larger circuits. Next we present a faster approach for solving the problem in a piecewise linear assumption.

## 3.1 Polynomial Time Linear Programming Approach

It has already been observed that the dependence of delay and leakage on thresh-hold follows a convex curve. This convexity property can be exploited to generate a linear program under the piecewise linear assumption. Piecewise linear approximation essentially signifies the replacement of the continuous curve by a piecewise linear curve. A similar approach was presented in [1] for the slack distribution in the placement problem.

Figure 1 shows a representative plot between thresh-hold and delay. This curve is convex in nature and is approximated by 3 lines as shown with parameters l1= (m1,c1), l2 = (m2,c2) and l3 = (m3,c3) (each line has a corresponding slope and constant offset). Let us consider a point (v2,d2) as shown in the figure. Since the curve is convex, the following property always holds

$$m2v2 + c2 \geq m1v2 + c1 \quad (8)$$

$$m2v2 + c2 \geq m3v2 + c2 \quad (9)$$

Hence a point ((v2,d2) in this case) which lines on a region of the curve approximated by a specific line (l2 in this case), has the highest value on that line. Hence the point (v1,d1) will have the highest value on line l1. A similar property could be illustrated for leakage current also. This property will be used heavily to define a linear programming formulation under the piecewise linear assumption.

### 3.1.1 Problem Formulation

Let us assume that each gate in the circuit has an associated delay vs thresh-hold curve for each input pin to gate output (equation 2) and leakage vs thresh-hold curve. As discussed before, leakage current in a gate also depends on the gate inputs since the inputs determine the gates which are turned off. In this work, we assume that the gate inputs in steady state have been determined already. If this state information is not available then a probabilistic approach in which input probabilities are used to determine the leakage current of the gate [5]. Note that even in this case the convexity property will not be violated (details omitted). For example, it has been shown that a NAND gate has three states as far as leakage is considered. These include three distinct sets of input combinations (0,0), (1,1) and (0,1),(1,0). The probabilistic leakage of a NAND gate is $I_{nand} = p_{0,0} * I_{nand}(0,0) + p_(1,1) * I_{nand}(1,1) + (p_{0,1} + p_{1,0}) * I_{nand}((0,1)/(1,0))$ where $p_{0,0}, p_{1,1}, p_{0,1}, p_{1,0}$ are the corresponding state probabilities. The computed curve could then be used in the optimization formulation. We assume each of these curves to be approximated by $m$ lines in a piecewise linear fashion. Let $z_g$ denote the leakage of a gate, let $a_g$ be the arrival time of a gate and $t_{i,g}$ denote the delay from pin $i$ to the output of $g$. Let the $i-th$ line representing the delay of $j-th$ pin for gate $g$ have the following parameters $y_i^{g,d,j}, c_i^{g,d,j}$. Here $y_i^{g,d,j}$ signifies the slope of the line and $c_i^{g,d,j}$ signifies the constant. Let the $i-th$ line representing the leakage curve for gate $g$ have the following parameters $y_i^{g,l}, c_i^{g,l}$.

$$Minimize \sum_{\forall g} z_g \quad (10)$$

$$z_g \geq y_1^{g,l} V_t(g) + c_1^{g,l} \quad (11)$$

$$z_g \geq y_2^{g,l} V_t(g) + c_2^{g,l} \quad (12)$$

$$-- \quad (13)$$

$$z_g \geq y_m^{g,l} V_t(g) + c_m^{g,l} \quad (14)$$

$$t_(i,g) \geq y_1^{g,d,i} V_t(g) + c_1^{g,d,i} \quad \forall input \quad pins \quad i \quad of \quad g \quad (15)$$

$$t_(i,g) \geq y_2^{g,d,i} V_t(g) + c_2^{g,d,i} \quad \forall input \quad pins \quad i \quad of \quad g \quad (16)$$

$$t_(i,g) \geq y_3^{g,d,i} V_t(g) + c_3^{g,d,i} \quad \forall input \quad pins \quad i \quad of \quad g \quad (17)$$

$$-- \quad (18)$$

$$t_(i,g) \geq y_m^{g,d,i} V_t(g) + c_m^{g,d,i} \quad \forall input \quad pins \quad i \quad of \quad g \quad (19)$$

$$a_g \geq t_{i,g} + a_i \forall input \quad pins \quad i \quad of \quad g \quad (20)$$

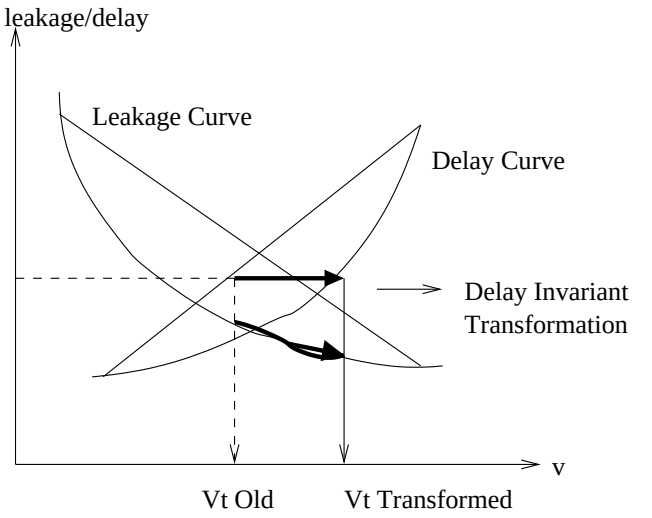


**Figure 2: Transformation to a Feasible Solution**

$$a_g = A_g \forall g \in PI \tag{21}$$

$$a_g \leq R_g \forall g \in PO \tag{22}$$

$$V_t - min \leq V_t(g) \leq V_t - max \forall g \tag{23}$$

Equation 10 illustrates the objective with the variable $z_g$ representing the leakage current for a gate assumed to follow a piecewise linear curve. Equations 11 to 14 represent the behavior of $z_g$. For all the $m$ lines used to represent the leakage curve, we need to find the particular line on which $z_g$ falls. These constraints along with a minimization objective will ensure that. Note that these constraints exploit the convex property of the leakage curve. These constraints would be defined for all gates. Equations 15 to 19 do something similar for the individual pin delays for each gate. Note that these equation once again will have to be written for all gates. For each of the $m$ lines on the linearized input pin delay curve for a gate, we are trying to find the the line on which the delay parameter $t_{i,g}$ lies. Equation 20 illustrates that the arrival time for a gate is maximum of the input arrival times plus the delay from input pin to gate output. Equation 21 assigns a pre-specified arrival time to primary inputs and equation 22 illustrates the required time constraint at the primary outputs.

This formulation will solve the leakage optimization problem optimally in polynomial time. It assumes a piecewise linear approximation for the delay and power curves.

### 3.1.2 Generating a Feasible Solution

The previous formulation solves the generalized problem where each gate could be assigned an independent thresh-hold voltage. This formulation generates assignments of thresh-hold voltages to gates considering a piecewise linear curve for leakage and delay. Now we transform this solution to the real curve. This transformation should be such that the generated solution should still be feasible from a delay constraint point of view. We call this transformation the

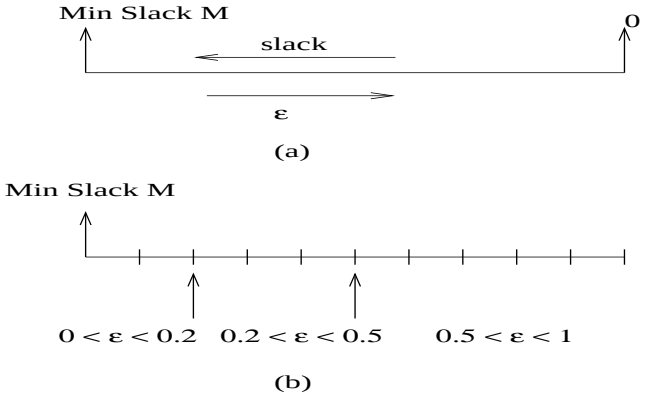


**Figure 3: Clustering Strategy**

*delay invariant* transformation which is illustrated in figure 2. Basically, the assignment of thresh-hold voltage to a gate corresponds to a certain gate delay given by a line on the piece-wise linear plot of delay vs voltage. As shown in the figure, keeping this delay fixed, we increase the thresh-hold voltage till we hit the real delay vs voltage curve. The thick horizontal arrow signifies this transform. Consequently, the leakage current of the gate decreases which is illustrated by the thick arrow on the leakage curve. Hence this transform ensures that the gate delay does not change along with a further reduction on the gate leakage current. This could be done independently for all the gates. Note that the discussed formulation solves both the thresh-hold selection and assignment problem simultaneously.

## 3.2 Fixed On-Chip Thresh-hold Voltages

The previous formulation assumed the independence of all gates as far as assignment of thresh-hold voltages is concerned. As mentioned before, this may not be economically a very viable option. Usually chips can have two thresh-hold voltages although having more than two thresh-holds may be an option the industry would consider in the near future. Let us assume that we can have $k$ distinct voltages on chip. The question we try to answer next is what should these $k$ thresh-hold voltages be, and which gates should be assigned which thresh-hold voltage. We extend the formulation described above to address this issue.

Let us assume, we know which gates will have the same thresh-hold voltage. Hence we have $k$ clusters of gates and all we have to determine is what these thresh-hold voltages should be. This can be easily enforced in the constraints presented above. Basically we can equate the thresh-holds of gates in the same cluster. This will result in a solution of upto $k$ distinct thresh-hold voltages and minimum possible leakage. Of course the next problem to consider is the generation of these cluster constraints. Also note that any general value of $k$ (including 2) can be easily considered by this generic mathematical formulation. Hence the possibility of having more than two thresh-holds can be easily investigated by our approach.

### 3.2.1 Iterative Gate Clustering

There are many ways in which gates could be clustered together. We propose a criticality driven iterative clustering technique. This is due to the observation that gates with similar criticality should be assigned the same thresh-hold

voltage. A gate is critical if it lies on the slowest path on the circuit. Following is the formal description of the strategy

1. Assign the lowest thresh-hold to all gates

2. Compute the slack for all gates (required time - arrival time). If some gates have positive slack then subtract a scaler quantity from all the gate slack such that all slack becomes negative with a range from M to 0. Here M is the most negative slack

3. Let us define an $\epsilon$-critical gate ($\epsilon \leq 1$) as a gate whose slack is within $\epsilon * M$. This means if a gate is $\epsilon$-critical then $slack(gate) \leq (1 - \epsilon) * M$

We will use $\epsilon$ as the parameter of clustering. Figure 3 illustrates this concept. Figure 3(a) illustrates the range of slack from 0 to Min-Slack $M$. The parameter $\epsilon$ measures the distance from Min-Slack M. Now let us suppose we have 3 distinct thresh-hold voltages and hence three clusters are desired. Our clustering strategy will generate results similar to figure 3(b). The solution represented in the figure clusters all the gates with slack within $0.2 * M$ in one partition, from $0.2 * M$ to $0.5 * M$ in another cluster and gates greater than $0.5 * M$ in the third cluster. This is based on the intuition that gates with similar criticality should have similar thresh-holds. The iterative strategy is as follows.

Let us discretize the $\epsilon$ range (0-1) into $n$ discrete points. These $n$ points signify the $n$ possible values of $\epsilon$. Given $i$ as the number of thresh-hold voltages allowable, we want to generate $i$ linear partitions of the line which represents $\epsilon$ (just like figure 3(b) where $i = 3$). We generate this partitioning as follows. Let us consider $i - 1$ possible locations where the $\epsilon$ range needs to be cut (to generate $i$ ranges). Let us assume $i = 2$, so we need to cut the range at one location. This range can be cut only at $n$ points. Hence we have (n-1) possible locations where this partition can be made. For each partition we figure out the gates that must be clustered together. The linear programming formulation is then executed iteratively for each partition and the partition that generates the minimum leakage is selected. Now if $i = 3$, then the $\epsilon$ range must be cut a two places (just like figure 3(b)). Once again the first cut can only be placed on $n$ locations. If the first cut is placed on location 1 on the range, then the second cut can be placed at $n - 1$ locations. If the first cut is placed at location 2 then the second one can be placed on $n - 2$ locations. Hence total number of combinations n*(n-1)/2, or $O(n^2)$. Among these partitions we select the one that generates the minimum possible leakage power. Generalizing this approach to $i$ thresh-hold voltages, the iterative complexity of this approach is $O(n^{(i-1)})$. Since $i$ is technology dependent, hence can be assumed a constant. So this approach should be polynomial in $n$ which can be controlled by the user.

Other heuristics could also be used to generate this clustering techniques. The formulation presented above is independent of how this clustering is generated. The presented approach solves the thresh-hold selection and assignment problem simultaneously. It can trivially be extended to multiple thresh-hold voltages (as compared to the approach by [7] which is good only for dual voltages). This can be used as an evaluation strategy for considering multiple thresh-holds in future. This completes the description of our algorithm.

### 3.2.2 Generating a Feasible Solution

Solving the linear programming formulation under clustering constraint poses another problem when transforming the solution from the piecewise linear assumption to the real curve. The approach for generating a feasible solution in the previous subsection (figure 2) will not work since we might end up increasing the number of distinct thresh-holds. One approach could be the *thresh-hold-invariant* approach in which we assume the solution given by linear program is the final solution. Of course this will also generate a valid feasible solution (in terms of delay constraint) but the leakage of the generated solution may be a little higher. Another approach could be to perform the *delay invariant* approach for all gates (figure 2). The thresh-hold of a cluster should then be decided by the lowest thresh-hold voltage gate in that cluster after *delay invariant* transformation.

## 4. RESULTS

We integrated the proposed algorithm in SIS. The standard cell library information in lib2.genlib along with the delay dependence equation 2 was used to generate delay vs thresh-hold curves for all gates. Leakage curves were generated similarly. All leakage curves were normalized w.r.t a basic inverter in the library. The MCNC benchmarks were optimized using standard scripts of SIS and mapped for minimum delay. The range of possible thresh-hold voltages for all gates was assumed to be 0.1 to 0.7. The delay constraint for the circuit was generated as follows. All gates were implanted using the lowest possible thresh-hold voltage. The resulting circuit delay with 5% slack was the constraint. The leakage and delay curves were then linearized with 8 lines. We used the *delay invariant* transformation for generating a feasible solution. Our experiments showed that this lead to a good tradeoff between runtime and quality. Experiments were then performed on MCNC benchmarks.

Table 1 illustrates the results. We experimented with one, two and four thresh-hold voltages. Comparisons were made with the optimal result (under the piecewise linear assumption) which assumes unbounded number of thresh-hold voltages. It can be seen that the optimal approach results in minimum leakage. One thresh-hold voltage results in largest leakage. Our formulation although does pick the best thresh-hold for all the gates. This is an improvement on traditional approach which will assign the lowest thresh-hold to all gates instead of selecting the best thresh-hold under the given constraints. Having two thresh-hold results in a massive improvement in leakage over one thresh-hold. Our formulation could also pick the best thresh-hold voltages for each design. Results for four thresh-holds are mixed in nature. For many benchmarks, four thresh-hold voltages does not give any significant improvement over two thresh-holds. But for $too-large, x4, rot, pair, apex6$ the improvements are non trivial. Hence for some designs having more than two thresh-holds could be an option the designer might want to consider. The presented algorithm/formulation could be used to make such decisions.

## 5. CONCLUSION

In this paper we presented a general mathematical framework for simultaneous thresh-hold selection and assignment. Using this approach the designers can automatically select the correct thresh-hold voltage for their design and also the

| Bench | Optimal | $1-V_t$ | | $2-V_t$ | | | $4-V_t$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Leak | Leak | $V_t-1$ | Leak | $V_t-1$ | $V_t-2$ | Leak | $V_t-1$ | $V_t-2$ | $V_t-3$ | $V_t-4$ |
| 9symml | 107.25 | 143.57 | 0.198 | 128 | 0.48 | 0.198 | 123.9 | 0.7 | 0.48 | 0.198 | - |
| C1355 | 282.0 | 350 | 0.15 | 328 | 0.37 | 0.15 | 319.8 | 0.7 | 0.47 | 0.376 | 0.15 |
| apex6 | 308.3 | 513.8 | 0.145 | 367.0 | 0.549 | 0.1 | 329.7 | 0.7 | 0.686 | 0.533 | 0.145 |
| apex7 | 109.24 | 163 | 0.2 | 123.2 | 0.7 | 0.2 | 123.1 | 0.7 | 0.44 | 0.2 | - |
| C1908 | 258.5 | 367.22 | 0.147 | 297.77 | 0.7 | 0.147 | 286.6 | 0.7 | 0.317 | 0.147 | - |
| C2670 | 317.4 | 494.4 | 0.173 | 349.5 | 0.64 | 0.173 | 338.6277 | 0.7 | 0.636 | 0.172 | - |
| C432 | 123 | 181 | 0.135 | 145.15 | 0.135 | 0.7 | 145.15 | 0.7 | 0.135 | - | - |
| C499 | 282 | 350.7 | 0.15 | 328.13 | 0.377 | 0.15 | 319.87 | 0.7 | 0.472 | 0.376 | 0.15 |
| alu2 | 222 | 327 | 0.155 | 258.35 | 0.154 | 0.7 | 248.44 | 0.7 | 0.549 | 0.44 | 0.155 |
| b9 | 59.44 | 90.33 | 0.16 | 65.26 | 0.7 | 0.16 | 62.79 | 0.7 | 0.647 | 0.437 | 0.16 |
| cordic | 33.32 | 45.13 | 0.197 | 38.8 | 0.63 | 0.197 | 37.16 | 0.7 | 0.63 | 0.485 | 0.197 |
| comp | 80.67 | 108.32 | 0.190 | 91.8 | 0.7 | 0.19 | 90.38 | 0.7 | 0.255 | 0.19 | - |
| pair | 623.3 | 983.9 | 0.152 | 720.4 | 0.6 | 0.152 | 677.7 | 0.7 | 0.6 | 0.31 | 0.512 |
| rot | 312.7 | 496.0 | 0.142 | 357.2 | 0.7 | 0.142 | 337.9 | 0.7 | 0.55 | 0.31 | 0.141 |
| x4 | 166.2 | 259.5 | 0.159 | 203.79 | 0.7 | 0.159 | 189.84 | 0.673 | 0.389 | 0.276 | 0.159 |
| too-large | 159.77 | 239.34 | 0.151 | 191.97 | 0.7 | 0.151 | 175.46 | 0.699 | 0.45 | 0.35 | 0.13 |

**Table 1: Results for many thresh-hold voltages**

specific gates for those thresh-holds. The algorithm can be trivially extended for considering more than two thresh-holds. Hence the algorithm provides a lot of flexibility to designers. An interesting course of future work could be to develop more elaborate and faster techniques of gate clustering.

# 6. REFERENCES


[1] " A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement ". In *IEEE Trans. on Computer Aided Design*, pages 1332–1341, Nov 1997.

[2] B.J. Sheu, D.L. Scharfetter, P.K. Ko and M.C. Jeng. "BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors ". In *IEEE Journal of Solid-State Circuits*, pages 558–566, Aug 1987.

[3] J. Kao, A. Chandrakasan and D. Antoniandis. "Transistor Sizing Issues and Tools for Multi-Thresh-hold CMOS Technology". In *Proc. Design Automation Conference*, June 1997.

[4] M. Anis, S. Areibi and M.I. Elmasry. "Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gat Clustering Technique". In *Proc. Design Automation Conference*, June 2002.

[5] Q. Wang and S.B.K. Vrudhula. "Static Power Optimization of Deep Submicron CMOS Circuits for DUAL Vt Technology". In *Proc. International Conference on Computer Aided Design*, pages 490–496, Nov 1998.

[6] R.X Gu and M.I. Elmasry. "Power Dissipation Analysis and Optimization of Deep Submicron CMOS Digital Circuits". In *Proc. IEEE Journal of Solid-State Circuits*, pages 703–713, May 1996.

[7] V. Sundararajan and K. Parahi. "Low Power Synthesis of Dual Thresh-hold Voltage CMOS VLSI Circuits". In *Proc. International Symposium on Low Power Electronics and Design*, 1999.