Improved Threshold Voltage Assignment via Combinatorial Implementation Selection

Soheil Ghiasi
Department of Electrical and Computer Engineering
University of California, Davis
soheil@ece.ucdavis.edu

ABSTRACT

We study the problem of sub-threshold leakage current optimization using dual threshold voltages under timing constraints. We develop an analytical framework that addresses the more general problem of gate implementation selection under timing constraints. While NP-Complete in the general case, we present conditions under which the problem of implementation selection from a library with discrete delay choices can be optimally solved in polynomial time. We also extend our methodology to handle situations that do not satisfy the theoretical conditions. Experimental results show that our algorithm reduces the leakage current by close to an order of magnitude, with no or negligible delay penalty. More importantly, our approach can quickly obtain near-optimal solutions for real life netlists. On average, our results are only 0.2% *leakier than the optimal solutions derived using MILP solvers, while our tool runs* 73 *times faster than GLPK*[1] *solver. Moreover, our algorithm outperforms the result of a recent LP-based competitor by* 33%.

1. INTRODUCTION

Power consumption is one of the most important quality metrics for digital systems due to its significant impact on density, battery life, robust operation and cooling costs. Traditionally, designers have been less concerned with active-state leakage current due to its negligible effect on total power consumption in previous technology nodes. However, as feature sizes continue to shrink, the contribution of leakage current to total power consumption grows significantly. Figure 1 illustrates the growing impact of sub-threshold leakage [2] across several technology nodes [7].

Exponential dependence of transistor leakage on its threshold voltage has motivated the idea of manufacturing the designs with two threshold voltages. In this approach, transistors (or standard cells) on the critical path of the design will have low threshold voltage to maintain their high performance operation, while some of non-critical transistors (or standard cells) are fabricated with high threshold voltage to improve design leakage under timing constraints. Although there has been several previous efforts to address threshold voltage assignment, or its combination with other design techniques such as dual V_{dd} and gate sizing [14, 20, 1, 13], to the best of our knowledge there has been no analytical study of optimality and power savings bounds in dual V_t technology. Hence, it is hard to know if existing algorithms fully exploit the potential benefits of design with two threshold voltages.

We study the problem of threshold voltage assignment as a special case of the general implementation selection problem (aka circuit implementation [27]). We significantly extend the existing results in time budgeting to perform near optimal implementation selection from a library of components with arbitrary delay choices. We prove that arbitrary delay choices can make the problem NP-Complete in the general case. Furthermore, we show that under reasonable assumptions implementation selection from a library of components with arbitrary delay choices can be solved efficiently, and in some cases optimally.

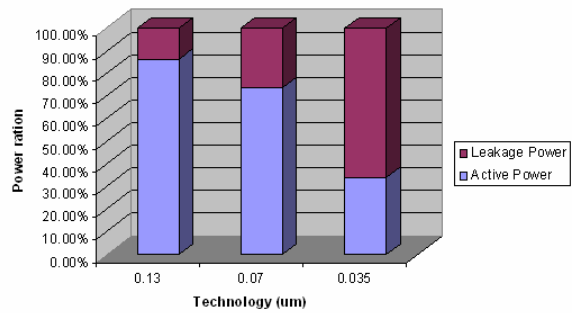


Figure 1: Estimation of active and leakage power variation across the technologies [7].

We apply our results to leakage optimization problem via threshold voltage assignment in dual V_t technology. This problem has been selected as the sample application domain due to its significance in future technology nodes. Experimental results show 76.6% and 82.4% reduction in leakage current of several MCNC benchmarks, with no or 5% delay penalty, respectively. Our algorithm outperforms a simultaneous V_t selection and assignment competitor by 33%, while we are only less than 1% away from the optimal solution obtained by solving the corresponding mixed integer linear programming (MILP) problem. Our algorithm is also extensible to V_t assignment in multi threshold voltage technologies. Although such technologies might not be presently economical, their power saving potentials can be simulated. We show that our technique can produce near-optimal solutions for multi-threshold technologies as well.

2. BACKGROUND

We adopt the standard representation of a gate-level netlist with a directed acyclic graph (DAG). Similarly, we use the conventional fanin, fanout, arrival time, critical path and timing constraint definitions for netlists [9]. Nodes of the DAG denote gates (or tasks at the system-level) and edges model the interconnects (or task depen-

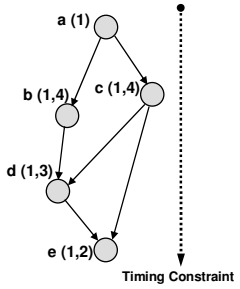
[1] GNU Linear Programming Kit

[2] Sub-threshold leakage is the dominant leakage mechanism. Consequently, throughout this paper we focus on sub-threshold leakage, and will use the terms leakage and sub-threshold leakage interchangeably.

dencies at the system-level). Each gate takes a specific amount of time to perform its computation (known as gate delay) and broadcast the result to all of its fanouts. The gate delay depends on the gate structure and the output load capacitance. We assume that all primary inputs arrive at time zero, and all primary outputs must be ready by a given timing constraint, denoted by T.

Edges are temporarily considered to have zero delay, however, note that the model can easily capture interconnect delays by insertion of dummy nodes with non-zero delays to edges. We assume that there are a number of different ways to implement each gate. For example, there might be several library elements that can realize a particular gate, e.g., implementation with high-threshold or low-threshold voltage. The problem of implementation selection is to select each gate from a given number of implementations, such that the timing constraint is met, and some objective function is optimized.

To accelerate the quality assessment of different solutions, it is reasonable to characterize the cost of each implementation separately, and estimate the solution cost to be the sum total of the selected nodes' costs. Furthermore, the cost of a particular implementation of a gate can be related to its delay. The estimation process transforms the problem of implementation selection to maximizing an easy-to-calculate function of gate delays, under timing constraint. In addition, for many practical applications, design variables are either integers or can be transformed to integer numbers with problem scaling. Therefore, it is often assumed that the problem arises in integral domain, i.e, initial gate delays, possible lower and upper bounds on gate delays, and timing constraint are non-negative integers.



a	1	1	1	1
b	1	1	1	4
c	1	1	1	4
d	1	1	3	1
e	1	2	1	1
Gain Function	5	6	7	11
Timing Constraint	4	5	6	7

Figure 2: System-level implementation selection with discrete delay choices. The table illustrates example optimal selections under several timing constraints.

Figure 2 illustrates some examples of implementation selection problem. Each node in the figure is annotated with its delay choices corresponding to different possible implementations. In this example, the gain function is assumed to be the sum of node delays, that is node costs are inversely proportional to their delays. Each column of the table shows an optimal implementation of the netlist subject to the given timing constraint. Note that the delay choices of nodes b, c and d are discrete integers, while the delay choices of nodes a and b are viewed as consecutive integers with proper lower/upper bounds. In practical applications, however, the gain associated with each implementation does not necessarily grow linearly with its delay.

In this paper, we discuss the problem of implementation selection under *discrete* integral delay choices. We leverage the existing techniques for implementation selection from a library with *continuous* integral delay choices to establish analytical bounds. Furthermore, we advance the state-of-the-art by developing polynomial-time algorithms that efficiently solve the problem for realistic non-linear gain-delay relations. We prove that our approach is effective for any convex relation between the gain associated with a gate and its delay. To validate our results, threshold voltage assignment in dual V_t technology has been selected as a sample application. However, our results are quite generic and applicable to many other gate-level implementation selection problems.

3. RELATED WORK

Implementation selection or its closely related counterpart, time budgeting, relate to the conventional slack distribution problem, which has been studied extensively in the synthesis community. However, almost all of the techniques developed in previous research efforts, and employed in industrial tools utilize sub-optimal heuristics with no guarantee on the solution quality. The majority of previous work have used Zero Slack Algorithm (ZSA) [16] or an extension of it, to assign timing constraints to components of a design. ZSA selects a path of the graph, assigns delay budget to the nodes on this path according to some criteria, and repeats this procedure until no further delay relaxation is possible. Such an approach is shown to be sub-optimal [8].

In our previous work, we presented a unified theoretical framework that optimally and in polynomial time solves several widely-used formulations of implementation selection under continuous integral delay choices [18]. However, the framework is incapable of addressing the problem under discrete delay choices. The problem under discrete delay choices has been investigated by several researchers. However, they mainly considered the problem from a particular application's point of view and have not presented any analytical results on their algorithm quality. We have showed that the solution can be approximated within any given bound when the application graph has the form of a rooted tree [19].

The problem of threshold voltage assignment for leakage optimization has been investigated by several researchers. Wang et al. [14] present an intuitive heuristic algorithm for V_t assignment. In [21] a technique for leakage optimization via simultaneous V_t selection and circuit sizing is presented. Khandelwal et al. focus on simultaneous V_t selection and assignment [24]. However, to the best of our knowledge all of the previous efforts utilize sub-optimal heuristics and do not analyze the optimality of their results. Our technique comes very close to the optimal, but exponentially expensive to compute, solution while running about two orders of magnitude faster on selected benchmarks.

From the application point of view, the increase in implementation delay at the task level (where nodes are not necessarily gates) has been utilized to improve system utility, optimization runtime and product time to market. Design timing closure [6, 12], voltage assignment [22], power optimization via gate and wire sizing [5, 4], high-level synthesis [11, 26], and software optimizations [10, 19] are only a few examples of the applications that have been considered by researchers.

4. SUBTHRESHOLD LEAKAGE AND DELAY MODEL

According to the BSIM model [3], subthreshold leakage current of a MOS transistor can be approximated as follows:

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

where $A = \mu oC_{ox}(W_{eff}/L_{eff})(kT/q)^2e^{1.8}$ and C_{ox} is the gate oxide capacitance per unit area and V_t is the threshold voltage. Equation 1 exhibits a exponential increase in the leakage current with increase of the threshold voltage. However, decreasing the threshold voltage would slow down the transistors. Specifically, delay of

a MOS transistor follows the following approximate relation with respect to V_t [25, 3, 17]:

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

where α is a technology dependent factor, which is around 1.3 for short channel and 2 for long channel devices.

The leakage of a CMOS gate depends on the number and topology of transistors that are turned off and hence on the inputs. For example the NMOS transistors of a NAND gate are in a stack, while its PMOS transistors are in parallel. Consequently, a NAND gate that has both NMOS transistors off (input = 00) has smaller leakage current compared to the case of having both PMOSs off (input = 11).

We model the leakage of a gate as a weighted average of its leakage under various input patterns. Thus, the input patterns that are more likely to occur contribute more to the average leakage of a gate. To estimate the delay, we simulated the delay of a gate over a range of output loads. Then, a linear fit is carried out to approximate the gate delay as a linear function of its structure and output load. These two models are rather standard and are widely used in the community.

In this paper, we use a lookup-table (LUT) based approach to store the average leakage, intrinsic delay and slope for the load dependent delay of gates. We lookup the parameters and quickly estimate the leakage and delay of each gate throughout the optimization process. Many other researchers have utilized similar LUT-based techniques, and have analyzed its decent accuracy, and accuracy-storage tradeoff [2, 20]. Leakage current of a CMOS circuit is calculated simply as sum total of the leakage currents of all gates. The leakage and delay entries of the lookup table are adopted from [24] to allow a fair comparison in Section 6.

5. LEAKAGE OPTIMIZATION VIA THRESHOLD ASSIGNMENT

5.1 Problem Transformation

We formulate the problem of gate-level V_t assignment in dual V_t technology as an instance of implementation selection with discrete delay choices. The two delay choices for a gate are its delay under low and high threshold voltages considering its load. The objective is to maximize the savings in leakage current, through assignment of selected gates to high threshold voltage, while meeting the timing constraint. We explain our methodology using dual implementation choices. In next subsection, we show that our approach is extensible to some of the cases where multiple implementations are present, including multi threshold voltage technologies.

In dual V_t technology, gates have exactly two delay choices, i.e., there exist exactly two possible implementation points in the "leakage savings-delay" plane for each gate. We temporarily relax the assumption of discrete integral delay choices. The two points can be assumed to form a line with a fixed slope. The slope represents the increase of the leakage savings by replacing an implementation with another implementation with additional unit delay. Note that neither the points on the line are valid implementations of the gate nor the relation between the gate leakage and its delay is linear. Nevertheless, the slope of the line forms an intuitive basis for leakage savings per unit delay increase of a gate.

Figure 3.d illustrates the situation. The two end points corresponding to $V_{t,low}$ and $V_{t,high}$ are the two possible implementations of a gate. The other points on the line segment are temporarily added to the implementation choices of the gate, and do not actually exist in the library. Hence, they cannot appear in the final solution. However, they allow us to temporarily assume that the implementations are selected from a library of continuous integral delay choices.

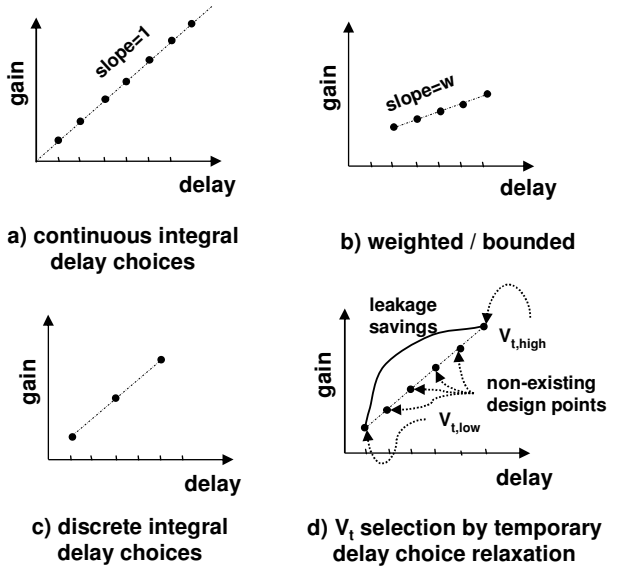


Figure 3: Implementation choices in the gain-delay plane. Case d) is temporarily relaxed to get an instance of case b).

The corresponding implementation selection from a library of continuous integral delay choices is to select the proper implementation for each node (from all of the points on the line segment) such that the total weighted delay is maximized under timing constraint. This corresponds to the case illustrated by Figure 3.b, which is a natural extension of the case with unbounded implementation choices and identical nodes (Figure 3.a). Note that the imaginary implementations have linear delay-leakage relation and hence, the total weighted delay objective function captures the total leakage savings. The delay-leakage relation is accurate at the two realistic implementation points, i.e, end points of the line

In the next subsection, we utilize our previous results to show that the temporarily relaxed problem (the case for Figures 3.a and .b) is optimally solvable in polynomial time. Then, we present properties of the delay choices, under which, a library with discrete delay choices (Figure Figures 3.c) can be transformed to continuous integral delay choices. In case a problem transformation is not possible, we round down the solution of relaxed problem to arrive at a feasible solution. For threshold voltage assignment problem, this means that all of the implementation choices between the two end points of the line segment will be assigned $V_{t,low}$ in the final solution.

We show that the proper assignment of weights guides the algorithm to select one of the two end points from implementation choices, so that almost none of the gates need rounding legalization. As a result, only a small amount of leakage savings is eliminated by rounding down the continuous integral solution.

5.2 Implementation Selection with Continuous Integral Delays

The design model that we considered so far, assumes that nodes incur some delay to perform their associated computations, and edges have zero delay. More generally, we can assume that edges incur some delay, and nodes have zero delay. A netlist with implementation choices for nodes can be transformed to an edge imple-

mentation instance by 1)splitting each node to two nodes that are connected by an internal edge, and 2) assigning proper weights to internal, and 3) zero weight to external edges. The transformation of Figure 4.a to 4.b depicts the idea. It follows that the problem of implementation selection for nodes is a special case of implementation selection for edges.

Previously, we have shown that the problem of edge implementation selection with continuous integral delay choices is the dual of a min-cost flow instance [18]. This problem aims to maximize the sum of weighted edge implementation delays in which, each edge of the DAG is associated with a constant weight. Moreover, the technique can handle lower and/or upper bounds on the delay of an edge. This corresponds to the case illustrated in Figure 3.b.

To summarize part of the previous result that relates to this paper, the primal edge implementation selection on a DAG formulates a dual min-cost flow problem on an extended DAG. The cost of the unit flow on edge e_{ij} of the DAG in the min-cost flow instance is the negative of the minimum delay choice possible for that edge. In addition, edges e_{ji} with cost u_{ij} are added to the original DAG, where u_{ij} is the upper bound of the delay choices of the edge e_{ij}. In the dual min-cost flow instance, the amount of flow supply at node i is equal to the difference of incoming and outgoing edge weights $(weight_{in}(i) - weight_{out}(i))$. The transformation of Figure 4.b to 4.c visualizes this process.

It follows that the primal problem (edge implementation selection) can be solved optimally in polynomial time, using any of the well-known min-cost flow algorithms [15]. Figures 5.a and 5.b illustrate the dual min-cost flow problem and its solution for an example graph whose edges have a lower bound of one and upper bound of three units of delay, and its timing constraint is four delay units. All of the edges are assumed to have identical unit weights.

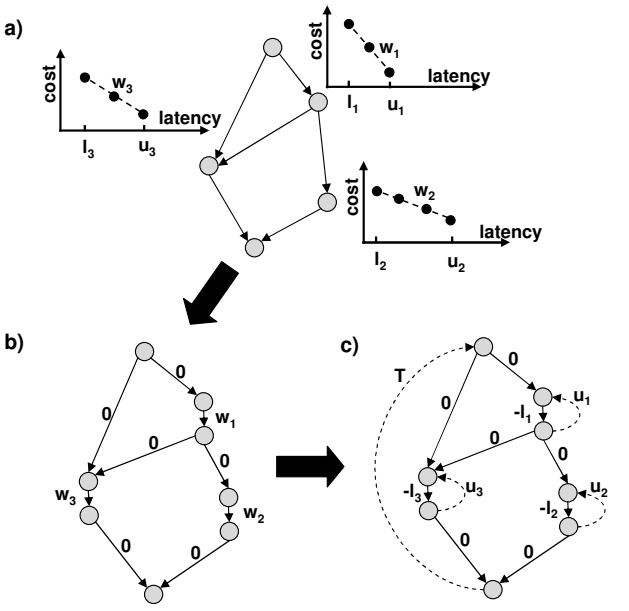


Figure 4: Implementation selection with continuous integral delays can be solved using its dual min-cost flow instance [18].

Once the flow variables along edges e_{ij} and e_{ji} are determined, we construct the residual graph. For any edge in the graph with non-zero flow along it, there are two forward and backward edges in the residual graph. The cost of each backward edge is negative of the forward edge cost. Let δ_i be the shortest distance of node i to *SO* in the residual graph. There is no negative cost cycles in the residual graph and hence, δ_i variables are well defined. δ_i variables can be determined by utilizing any well-known shortest path algorithm, such as Bellman-Ford [23], that is applicable to graphs with negative edge costs. The proper implementation of edge e_{ij} is determined by selecting the implementation whose delay is equal to $\delta_j - \delta_i$. Figure 5.c shows the optimal solution for the example graph in which, edges are annotated with their implementation delay.

Theorem 1. *$\delta_j - \delta_i$ forms an optimal implementation for edge e_{ij}, where δ_i is the shortest path of node i to SO in the residual graph. The residual graph is formed by solving the aforementioned min-cost flow instance.*

Proof: Based on the LP duality theorem and complementary slackness condition. Details are omitted for brevity [18, 15].■

The time complexity of the algorithm is determined by the min-cost flow step, which functions on the extended graph with $O(n)$ nodes. Therefore, the min-cost flow instance can be solved in $O(m.log(n).(m + n.log(n)))$ via enhanced capacity scaling algorithm [15], where m is the number of edges in the graph. For practical CAD problems, the graphs are usually sparse in which case, the degree of nodes is bound by a small constant. Thus, the number of edges is $O(n)$ and the time complexity is reduced to $O(n^2.log^2(n))$, which is quite affordable for many problem instances. Section 6 reports the experimental observations including runtime measurements of our approach. The experimental results verify the fast runtime of our algorithm on commercial problem instances, and advocate its practicality.

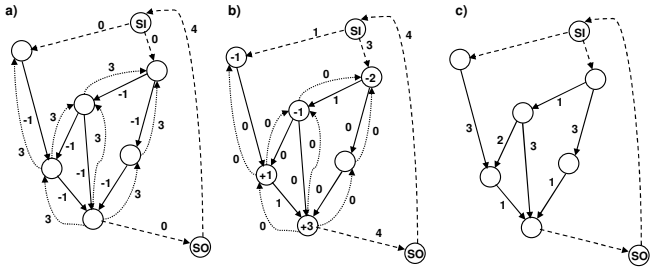


Figure 5: a)The Min-cost flow instance of an example edge implementation selection. b)Min-cost flow solution. Flow supply at node i is $weight_{in}(i) - weight_{out}(i)$. c)Optimal solution with maximum total weighted delay.

5.3 Threshold Voltage Assignment via Implementation Selection with Discrete Delays

In this section, we remove the relaxation assumption of continuous integral delay choices for each gate, and focus on discrete delay budgeting. We present our results on the complexity of the problem and conditions, under which, the problem can be solved optimally.

If each node has arbitrary delay choices, the problem of implementation selection to maximize total delay is NP-complete. Even under very restrictive assumptions of two arbitrary delay choices for each node, and an application graph in the form of a path, the problem is NP-complete.

Theorem 2. *The problem of implementation selection from a library of discrete delay choices for maximization of total delay is NP-complete.*

Proof: We reduce a given instance of subset sum problem to an instance of implementation selection on a path. For each element e_i of the set, we add a node with two delay choices of zero and

e_i to the path. The question of "is there a subset of the set such that the total value of elements in the subset is equal to B?" can be answered in polynomial time if the corresponding implementation selection problem on the constructed path, with timing constraint B, can be solved in polynomial time. Therefore, the implementation selection problem with arbitrary delay choices is NP-complete. ■

Although the implementation selection problem is NP-complete in the general case, a special relation between the delay choices, or the structure of the graph, allows efficient solution of the problem. For example, under arbitrary delay choices, the optimal solution can be approximated to any given accuracy if the graph forms a rooted tree [19]. We show that the problem can be efficiently solved for the general graph structure under some special relation between the node delay choices.

Let us assume that the delay choices are consecutive multiples of an integer d. That is, the delay choices of a given node are $m.d, (m+1)d \ldots (m+n).d$ for given integers m and n, and the timing constraint is T. The following lemma proves that the aforementioned instance of the problem can be transformed into implementation selection from continuous integral delay choices, and hence, can be optimally solved through the technique explained in Subsection 5.2:

Lemma 3. *The implementation selection problem instances, and solutions are scalable by an arbitrary positive integer. That is, an optimal integral solution for a given problem instance, if multiplied by d, forms an optimal solution for a new discrete implementation selection instance. The new instance is created by multiplying delay choices, lower bounds, upper bounds and timing constraint by d.*

It follows that the aforementioned instance can be scaled down by the factor d to create an implementation selection instance under continuous integral delay choices with lower and upper bounds of m and n for the given node, respectively. The timing constraint is reduced to $\lfloor T/d \rfloor$. Note that the timing constraint does not necessarily have to be an integral multiple of d.

For the case of V_t assignment, the two delay choices, and their relation completely depends on the choice of threshold voltages. Although threshold voltages might be selected to enable exact scaling, it should not be generally assumed that they follow this pattern. Therefore, we solve the threshold voltage assignment problem by relaxing it into implementation selection from continuous integral delay choices, followed by solution round down to the closest feasible implementation for each gate. This scheme works very effectively in practice, and most of the gates with enough timing slack are assigned to high threshold voltage.

In case of multi threshold voltages, more than two implementations are available for each edge [3]. We observe that the leakage current exponentially decreases with increase of V_t. Hence, the gain slopes determining the edge weights, decrease with addition of implementation choices for each gate. Figure 6 illustrates the case, and shows how we can transform it to a conventional implementation selection instance. The partial graph on the right part of Figure 6 replaces one of the split nodes shown in Figure 4.b. This technique is extensible to piece-wise linear approximation of cost functions, and finds the optimal solution (or near-optimal solution for discrete case) via purely combinatorial algorithms. Note that similar linear programming based methods employ numeric optimizations, which exhibit slow runtimes and occasional numerical instability.

[3]Recall that nodes are modeled using edges by node splitting (Figure 4)

It is noteworthy that the condition $w_1 >= w_2$ guarantees that starting with the implementation A, limited available slack would be spent to select the implementation B rather than C. In the general piece-wise linear approximation scheme, the condition $w_i >= w_{i+1}$ implies the convexity of the cost function, which is a necessary condition for optimality of our analysis in the continuous domain. The transformation does not work if weight is not a non-increasing function of V_t in which case, the cost function will not be convex. The continuous integral version of the problem is NP-Complete in that case.

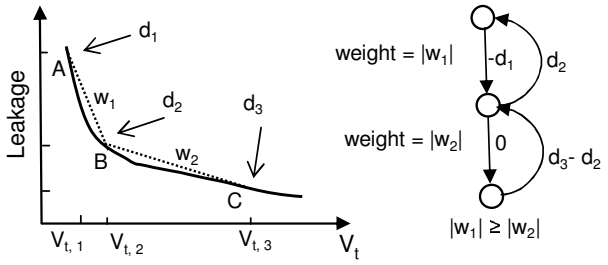


Figure 6: Handling multi threshold voltages by transformation to a conventional implementation selection instance.

6. EXPERIMENTAL RESULTS

We implemented our min-cost flow based algorithm for solving the relaxed implementation selection from a library of continuous integral delay choices in SIS [9]. Since the relaxed solution might not be feasible in dual V_t technology, we perform round down on the relaxed solution to arrive at a feasible discrete budgeting solution.

The round down process assigns $V_{t,low}$ as the threshold voltage of any gate whose delay is determined to be less than delay under $V_{t,high}$, in the relaxed version of the problem. That is, all of the design points between two valid implementation will be realized using the faster implementation in order to meet the timing constraint. We used the experimental framework developed by Khandelwal et al. [24] to estimate the gate leakage and delay variations with respect to V_t for gates in lib2.genlib library. Similar to their approach, all leakage and delay curves are normalized with respect to a basic inverter in the library.

The MCNC benchmarks are mapped to the lib2.genlib library under default settings. In one set of experiments, the timing constraint is set to the netlists critical path under $V_{t,low}$. In another experiment 5% relaxation is allowed to this timing constraint. The $V_{t,low}$ and $V_{t,high}$ values are adopted from [24] to allow a fair comparison. The threshold values depend on the benchmark and vary in the range of 0.3 to 0.5 volts. Nevertheless, our algorithm and approach is quite generic and applicable to other threshold values.

After assigning low and high threshold voltages to gates, another timing analysis is performed to assure the validity of results. In all cases, our algorithm did meet the required timing constraint. For each benchmark, the netlists are processed without applying any SIS optimization. They are mapped using the built-in "map" command with no switches. The choice of benchmarks, SIS optimization and mapping switches have been merely duplicated from [24] to maintain consistency, and create a fair ground for comparison.

Table 1 summarizes our experimental results for dual V_t technology. The first column of the table illustrates the selected benchmarks. Second and third columns of the table show the value of low and high threshold voltage for each benchmark. We also re-

peated the same experiment with three threshold voltage whose results are reported in Table 2. Although fabrication processes with three threshold voltages are not economical today, we performed this experiment to demonstrate the effectiveness and extensibility of our methodology, in case such processes become viable in the future.

The "Initial leakage" column shows the leakage of benchmarks when all of the gates are fabricated with low threshold voltage ($V_{t,low}$). The next three columns report the result of applying "simultaneous V_t selection and assignment" [24], our min-cost flow based (MCF-based) algorithm with tight timing constraint, and our algorithm with 5% relaxation in the timing constraint. In each case, we report the optimal results obtained from solving the corresponding MILP problem instance. Note that MILP formulations are NP-hard in general, and finding the optimal solution by solving the MILP instance is inefficient for large problem instances. However, we report the optimal solutions to show the limits on power savings using dual threshold voltages.

Column "Simultaneous" was calculated with 5% relaxation in the timing constraint (5% delay penalty). Hence, our algorithm results with relaxed timing constraint should be considered for fair comparison. The comparison among normalized leakage current of the four methods is visualized in Figure 7. For each benchmark, initial leakage, optimized leakage after application of "simultaneous V_t selection and assignment", our min-cost flow based results and the optimal MILP-based results are illustrated. The figure depicts the substantial leakage savings of MCF-based method over baseline and "simul" algorithm results. More importantly, it shows that our result are very close to the best feasible solution.

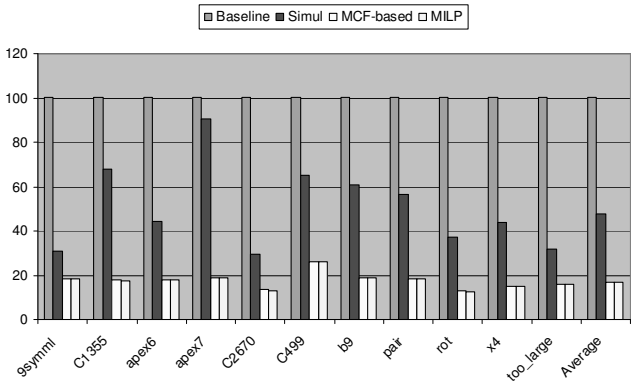


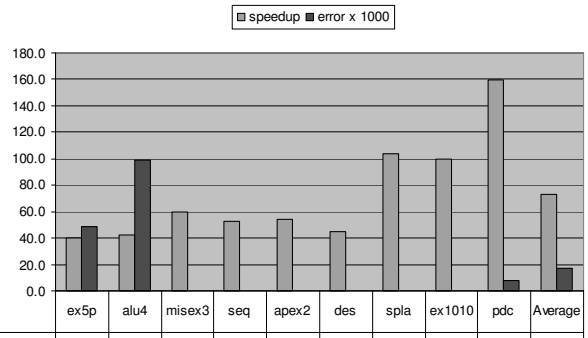
Figure 7: Comparison of normalized leakage current with 5% delay penalty. Our algorithm (MCF-based) comes very close to the optimal solution obtained by MILP solvers.

Under no delay penalty, the reduction in leakage current of the selected benchmarks is as high as 85% and, 76.6% on average. The improvement can be further magnified with extra relaxation in delay constraint. When 5% relaxation in timing is tolerated, the reduction in leakage is 82.4% on average, for the selected benchmarks. With the same timing constraint, simultaneous V_t selection and assignment [24] reduces the leakage 49.3%, compared to baseline implementation of assigning all of the gates to $V_{t,low}$. If the result of simultaneous V_t selection and assignment is considered as the basis for comparison, our algorithm improves the leakage by factors of 50.2% and 62.2% under no delay penalty, and 5% delay penalty, respectively. The sub-columns titled as "vs. init" and "vs. simul." illustrate the comparison of the corresponding results, with initial leakage, and leakage after applying simultaneous V_t selection and assignment. Our algorithm comes extremely close to the optimal solution, by generating results that are occasionally optimal, and on average only 0.2% worse than the optimal MILP solution.

The MILP solvers are known to have slow runtimes. Due to exponential dependence of solution space to problem size, MILP optimization runtime particularly becomes problematic for complex netlists. The testbenches used in the aforementioned set of experiments have at most a thousand gates for which, the runtime of our algorithm and MILP solvers are both negligible (on the order of tenths of a second for most cases). Note that the choice of testbenches was rather imposed to us to allow a fair comparison with a recent competitor [24].

In order to highlight the runtime improvement of our algorithm over MILP solvers and further evaluate its quality, we performed another set of experiments with nine of larger MCNC benchmarks. These testbenches are roughly one order of magnitude larger than the netlists of Table 1.

We summarize the important observations of the experiment in Figure 8. The figure shows that our algorithm is on the average 73 times faster than MILP solver for the selected benchmarks. The speedup was as large as 160 times for *pdc*, and as expected, it increased with the growth of the netlist complexity. Moreover, we were able to find the optimal MILP solution in most of the cases, and came extremely close (0.02% deviation on average) to the solution of MILP solver.



	ex5p	alu4	misex3	seq	apex2	des	spla	ex1010	pdc	Average
speedup	40.0	42.3	60.0	52.9	54.1	44.8	103.5	100.0	159.9	73.1
error x 1000	48.5	98.8	0.0	0.0	0.0	0.0	0.0	0.0	7.8	17.2

Figure 8: Error percentage and speedup comparison between MCF-based and optimal MILP solutions. The error percentage is multiplied by 1000 for better visualization.

7. CONCLUSIONS

We presented an analytical framework to address several versions of the implementation selection problems. We examined conditions under which, the implementation selection from a library of discrete delay choices can be optimally or very efficiently solved.

Under such conditions, the problem is transformed to an instance of implementation selection with continuous integral delay choices for maximization of total weighted delay. A simple rounding procedure might be required to legalize the intermediate solution. We showed that for practical CAD problems and with proper assignment of weights, the error introduced by rounding is negligible.

We leveraged the developed methodology and applied it to the problem of threshold voltage assignment in dual V_t technology. Our technique reduces the design leakage current by 76.6% or 82.4%, with no or 5% relaxation in timing constraint, respectively. It outperforms a recent simultaneous V_t selection and assignment technique by 33%. More importantly, our results are very close (less than 0.2%) to the optimal solution obtained by MILP solvers, while our algorithm ran about two orders of magnitude faster than GLPK MILP solver.

Benchmark	V_{t_l}	V_{t_h}	Initial leakage	Simul. leakage	Tight timing constraint				5% relaxed timing			
					MILP	leakage	vs. init	vs. simul.	MILP	leakage	vs. init.	vs. simul
9symml	0.40	0.50	515.70	159.24	132.3	132.3	74.35	16.92	94.86	94.86	81.61	40.43
C1355	0.40	0.50	2913.40	1985.08	848.3	883.4	69.68	55.50	503.75	521.28	82.11	73.74
apex6	0.41	0.50	3870.40	1720.33	746.6	746.6	80.71	56.60	692.60	698.41	81.96	59.40
apex7	0.41	0.50	1235.10	1116.24	290.0	290.0	76.52	74.02	231.19	233.12	81.13	79.12
C2670	0.38	0.49	6440.30	1885.21	1008.3	1022.3	84.13	45.77	848.84	865.79	86.56	54.07
C499	0.40	0.47	3062.60	1989.22	982.6	986.4	67.79	50.41	801.23	803.34	73.77	59.62
b9	0.41	0.50	590.50	357.61	142.1	142.1	75.94	60.26	110.42	110.42	81.30	69.12
pair	0.41	0.50	7014.71	3954.37	1373.5	1387.4	80.22	64.91	1268.20	1268.20	81.92	67.93
rot	0.39	0.50	4585.10	1692.89	686.6	686.6	85.03	59.44	585.12	596.07	87.00	64.79
x4	0.40	0.50	2881.30	1267.31	999.1	999.1	65.32	21.16	423.56	423.56	85.30	66.58
too_large	0.40	0.50	2413.50	768.16	408.5	408.5	83.07	46.82	385.38	385.38	84.03	49.83
Average			3229.33	1535.97	692.54	698.61	76.61%	50.17%	540.47	545.49	82.42%	62.24%

Table 1: The experimental results for design with two threshold voltages

Benchmark	V_{t_1}	V_{t_2}	V_{t_3}	Initial leakage	Simul. leakage	Tight timing constraint				5% relaxed timing			
						MILP	leakage	vs. init	vs. simul.	MILP	leakage	vs. init.	vs. simul
9symml	0.40	0.46	0.50	515.7	137.54	117.3	117.8	77.16	14.35	86.4	88.3	82.88	35.80
C1355	0.40	0.42	0.50	2913.4	1812.07	761.9	783.5	73.11	56.76	455.5	462.8	84.11	74.46
apex6	0.39	0.42	0.50	3870.4	1539.98	722.6	734.3	81.03	52.32	693.8	693.8	82.07	54.95
apex7	0.39	0.42	0.50	1235.1	777.9	256.5	258.1	79.10	66.82	228.5	230.1	81.37	70.42
C2670	0.37	0.46	0.50	6440.3	1545.1	810.5	813.6	87.37	47.34	668.4	669.3	89.61	56.68
C499	0.37	0.47	0.48	3062.6	1595.34	778.6	786.1	74.33	50.73	666.5	667	78.22	58.19
b9	0.40	0.42	0.50	590.5	308.72	136.9	138.4	76.56	55.17	109.2	109.2	81.51	64.63
pair	0.39	0.41	0.50	7014.7	3236.29	1343.2	1363.1	80.57	57.88	1261.5	1276.4	81.80	60.56
rot	0.38	0.41	0.50	4585.1	1194.89	632	632	86.22	47.11	567.5	569.5	87.58	52.34
x4	0.39	0.41	0.50	2881.3	1043.98	431.6	431.6	85.02	58.66	423.1	423.1	85.32	59.47
too_large	0.38	0.41	0.50	2413.5	527.19	408	413	82.89	21.66	371.5	371.5	84.61	29.53
Average				3229.33	1247.18	581.74	588.32	80.30	48.07	502.90	505.55	83.55	56.09

Table 2: The experimental results for design with three threshold voltages

8. ACKNOWLEDGMENTS

The authors would like to thank Dr. A. Srivastava and Mr. V. Khandelwal from University of Maryland for sharing their experimental setup and fruitful discussions that made this work possible.

9. REFERENCES

[1] A. Srivastava, D. Sylvester, D. Blaauw. "Concurrent Sizing, Vdd and Vth Assignment for Low-Power Design". In *Design Automation and Test in Europe Conference (DATE)*, pages 718–719, 2004.

[2] A.K. Sultania, D. Sylvester, S. Sapatnekar. "Tradeoffs Between Gate Oxide Leakage and Delay for Dual Tox Circuits". In *Design Automation Conference (DAC)*, pages 761–766, 2004.

[3] B.J. Sheu, D.L. Scharfetter, P.K. Ko, M.C. Jeng. "BSIM: Berkeley short-channel IGFET model for MOS transistors". *IEEE Journal of Solid-State Circuits*, 22(4):558–566, 1987.

[4] C. Chen, M. Sarrafzadeh. "Power Reduction by Simultaneous Voltage Scaling and Gate Sizing". In *Asia South Pacific Design Automation Conference*, pages 333–338, 2000.

[5] C. Chen, X. Yang, M. Sarrafzadeh. "Potential Slack: An Effective Metric of Combinational Circuit Performance". In *ACM/IEEE International Conference on Computer-Aided Design*, pages 198–201, 2000.

[6] C. Kuo and A. C.H. Wu. "Delay Budgeting for a Timing-Closure-Design Method". In *ACM/IEEE International Conference on Computer-Aided Design*, pages 202–207, 2000.

[7] D. Duarte, N. Vijaykrishnan, M.J. Irwin, H.S. Kim, G. McFarland. "Impact of Scaling on The Effectiveness of Dynamic Power Reduction Schemes". In *International Conference on Computer Design*, pages 382–387, 2002.

[8] E. Bozorgzadeh, S. Ghiasi, A. Takahashi and M. Sarrafzadeh. "Optimal Integer Delay Budgeting on Directed Acyclic Graphs". In *Design Automation Conference*, pages 920–925, June 2003.

[9] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, A.L. Sangiovanni-Vincentelli. *SIS: A System for Sequential Circuit Synthesis*. Memorandum No. UCB/ERL M92/41, Department of EECS. UC Berkeley, May 1992.

[10] F. Xie, M. Martonosi, and S. Malik. "Intraprogram dynamic voltage scaling: Bounding opportunities with analytic modeling". *ACM Transactions of Architecture and Code Optimization*, 1(3):323–367, 2004.

[11] J. Luo and N. Jha. "Battery-Aware Static Scheduling for Distributed Real-Time Embedded Systems". In *IEEE/ACM Design Automation Conference*, 2001.

[12] O. Coudert. "Timing and Design Closure in Physical Design Flows". In *IEEE International Symposium on Quality Electronic Design*, 2002.

[13] P. Pant, R.K. Roy, A. Chatterjee. "Dual Threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits". *IEEE Transactions on Very Large Scale Integration Systems*, 9(2):390–394, 2001.

[14] Q. Wang, S. Vrudhula. "Static power optimization of deep submicron CMOS circuits for dual VT technology". In

International Conference on Computer-Aided Design, pages 490–496, 1998.

[15] T. Magnanti R. Ahuja and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[16] R. Nair, C. Berman, P. Hauge and E. Yoffa. "Generation of Performance Constraints for Layout". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8:860–874, August 1989.

[17] R.X. Gu, M.I. Elmasry. "Power Dissipation Analysis and Optimization of Deep Submicron CMOS Digital Circuits". *IEEE Journal of Solid-State Circuits*, 31(5):707–713, 1996.

[18] S. Ghiasi, E. Bozorgzadeh, S. Choudhury, M. Sarrafzadeh. "A Unified Theory of Timing Budget Management". In *International Conference on Computer-Aided Design*, pages 653–659, 2004.

[19] S. Ghiasi, K. Nguyen, E. Bozorgzadeh, and M. Sarrafzadeh. "On Computation and Resource Management in Networked Embedded Systems". In *International Conference on Parallel and Distributed Computing and Systems*, pages 445–451, 2003.

[20] S. Sirichotiyakul , T. Edwards , C. Oh , R. Panda , D. Blaauw. "Duet: An Accurate Leakage Estimation and Optimization Tool for Dual-Vt Circuits". *IEEE Transactions on Very Large Scale Integration Systems*, 10(2):79–90, 2002.

[21] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda, D. Blaauw. "Stand-by Power Minimization Through Simultaneous Threshold Voltage Selection and Circuit Sizing". In *Design Automation Conference (DAC)*, pages 436–441, 1999.

[22] S.Raje, and M. Sarrafzadeh. "Scheduling with multiple voltages". *Integration*, 23(1):37–59, 1997.

[23] R. Rivest T. Cormen, C. Leiserson. *An introduction to algorithms*. MIT Press, 1990.

[24] V. Khandelwal, A. Davoodi, A. Srivastava. "Simultaneous Vt Selection and Assignment for Leakage Optimization". *IEEE Transactions on Very Large Scale Integration Systems*, 13(6):762– 765, 2005.

[25] V. Sundararajan, K. Parhi. "Low power synthesis of dual threshold voltage CMOS VLSI circuits". In *International Symposium on Low Power Electronics and Design*, pages 139–144, 1999.

[26] W. Zhang, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, D. Duarte, and Y.Tsai. "Exploiting VLIW Schedule Slacks for Dynamic and Leakage Energy Reduction". In *ACM/IEEE International Symposium on Microarchitecture*, 2001.

[27] W.N. Li, A. Lim, P. Agrawal, S. Sahni. "On the Circuit Implementation Problem". In *Design Automation Conference (DAC)*, pages 478–483, 1992.