An Effective Combinatorial Algorithm for Gate-Level Threshold Voltage Assignment

Soheil Ghiasi

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

Address:

University of California, Davis

Department of Electrical and Computer Engineering

One Shields AvenueDavis, CA 95616

Office:(530)752-0836

Fax:(530)752-8428

Email: soheil@ece.ucdavis.edu

Date of Receiving:

Date of Acceptance:

An Effective Combinatorial Algorithm for Gate-Level Threshold Voltage Assignment

Soheil Ghiasi

Abstract — *We study the problem of sub-threshold leakage current optimization using dual threshold voltages under timing constraints. We present time budgeting as a methodology to perform implementation selection through which, we assign design components to specific threshold voltages to maximize leakage savings. We discuss several well-known formulations of time budgeting, and present a theoretical approach that can optimally and in polynomial time solve those problems under consecutive integer delay choices. This technique is further extended to perform very efficient implementation selection from a library of components with arbitrary integer delay choices. Experimental results show that our algorithm reduces the leakage current by close to an order of magnitude, with no or negligible delay penalty. Our algorithm outperforms the leakage savings of a recent LP-based competitor by* 33%. *More importantly, our results approach the theoretical limits of leakage savings using dual and multiple threshold voltages that are derived using Mixed Integer Linear Programming (MILP) formulation of the problem. On average, our results are only* 0.74% *worse than the optimal solutions, while our algorithm runs about* 73 *times faster than GNU MILP solver.*

Keywords — Algorithm, Logic Synthesis, Leakage, Delay Budgeting, Implementation Selection, Slack Distribution, Threshold Voltage Assignment.

1 Introduction

Power consumption is one of the most important design 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 leakage power due to its negligible effect on total power consumption in previous technology nodes. However, as transistor sizes continue to shrink, the share of leakage current in total power consumption increases. Figure 1 illustrates the growing impact of sub-threshold leakage [1] in future technology nodes [9].

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 assigned high threshold voltage to improve design leakage under timing constraints [29]. The impact of this approach heavily relies on the efficiency of the threshold voltage assignment algorithm. Existing algorithms often employ path-based heuristics to select a subset of non-critical components. However, there has been no study of bounds on power savings using dual V_t technology, and hence, it is hard to know if existing algorithms fully exploit the potentials.

In this paper, we study the problem of threshold voltage assignment as a special case of the general implementation selection problem [40, 1]. We utilize the existing results in time budgeting to perform optimal implementation selection from a library of components with consecutive integer delay choices [33]. We show that arbitrary integer delay choices can make the problem NP-Complete. However, under reasonable assumptions time budgeting for arbitrary integer library choices can be solved efficiently, and in some cases optimally. We present a generic implementation selection algorithm, based on an intuitive extension of the implementation selection for consecutive integral delay choices, and apply it to gate-level threshold voltage assignment problem in dual V_t technology.

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 0.74% away from the optimal solution obtained by exhaustively solving the corresponding mixed integer linear programming (MILP) instances. Furthermore, our algorithms runs about 99 times faster than GNU Linear Programming Kit (GLPK) software that was used to solve the MILP instances. Our algorithm is also extensible to V_t assignment in multi threshold voltage technologies. Although such technologies are not presently economical, they might be practical in future. We

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

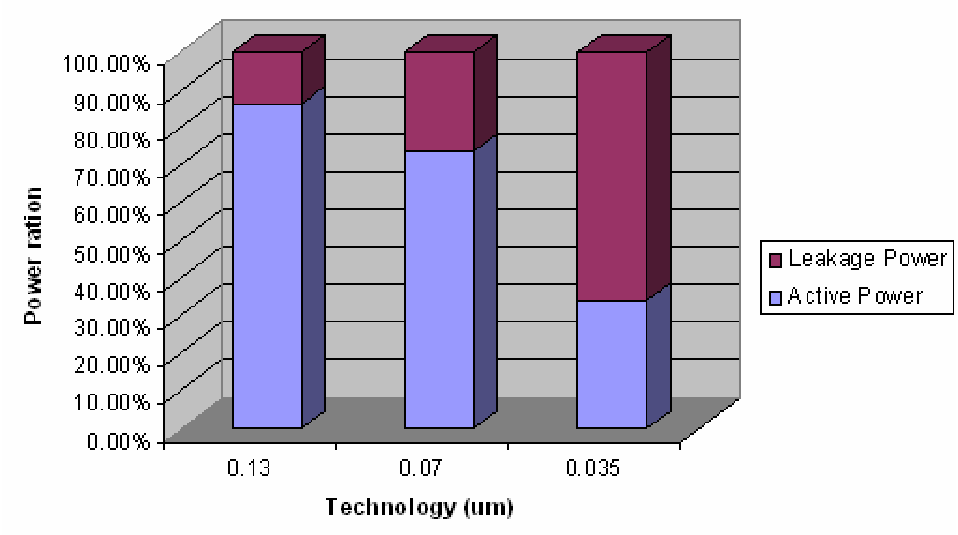


Figure 1: Estimation of active and leakage power variation across the technologies [9].

show that our technique is theoretically extensible to multi threshold technologies, and it can deliver very efficient solutions for such future technologies as well.

We proceed to present the execution model, assumptions and some background on time budgeting problem in Section 2. Section 3 reviews the existing results on the problem and discusses several applications of time budgeting in different areas of hardware and software design. Section 4 presents the background information and problem of leakage optimization in dual V_t technology. We discuss our algorithm and experiments for assigning threshold voltage to gates of a design in Sections 5 and 6, respectively. Section 7 concludes the paper.

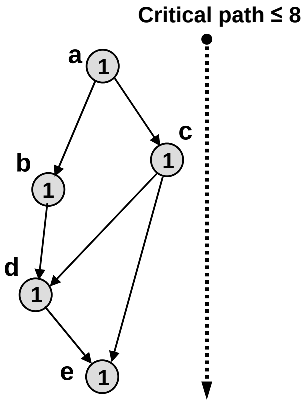
2 Background

In this section, we explain the definitions, application model, and other necessary background information that are used throughout the paper. We proceed by momentarily departing from gate-level netlists, and focusing on the generic implementation selection problem for application task graphs. In future sections, we will apply the developed model to gate-level threshold voltage assignment.

The left part of figure 2 illustrates an example of the application model used in this paper. We use the standard task graph model that can be represented as a directed acyclic graph (DAG). The nodes denote computations (or gates at the gate-level) and edges model the dependency among them. Each task starts when all its input data are available, i.e, all of its predecessors have finished their computations, and takes a specific amount of time to perform its computation and broadcast the result to all of its fanouts.

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 considered to have zero delay, however, this model can indeed account for edge delays by insertion of dummy data communication nodes on edges. The problem of timing budget management is to determine a local timing constraint for implementing each of the computations, such that the local constraint is feasible (not smaller than the minimum possible computation delay), the application timing constraint is met, and some objective function is optimized.

Example objective functions are power, area, optimization runtime, application quality (for example accuracy in object tracking or distortion in image compression), and testability. Most of such design quality metrics are difficult, if possible, to be estimated accurately at the application level. One reasonable solution is to characterize or estimate the utility improvement of each component with its delay relaxation. For example, the area of a component might be estimated to semi-linearly decrease with increase of its delay within some range. Figure 3 illustrates the relation of area vs. delay for 16-bit sequential multipliers of Xilinx CoreGen library [19] that justifies an example of such estimations. The estimation process transforms the problem of time budgeting to maximizing a simple function of local delays, under global timing constraint.



	Maximum Budget	Bounded Maximum Budget	Min-Max (within 25% of max)	Min-Max (within 40% of max)
a	0	1 (≥1)	1	1
b	4	3 (≤3)	2	1
c	4	3 (≤3)	2	1
d	0	0	1	1
e	0	0	0	1
Total Budget	8	7	6	5

Figure 2: An example of the execution model and some timing budget management policies. The table shows the optimal solution for maximum, bounded and min-max delay budgeting.

For many practical applications, design variables are either integers (such as delays in terms of number of clock cycles) or can be transformed into integer numbers with problem scaling. Therefore, it is reasonable to assume that the problem arises in integral domain, i.e, initial component delays, possible lower and upper bounds on local delays, and local and global timing constraints are non-negative integers.

Depending on the application domain, different time budgeting objectives are appropriate. An intuitive budget assignment policy tries to maximize the total delay budget assigned to the graph nodes, assuming that larger total budget correlates to larger improvements in the design utility. Another popular policy is to distribute the budget values fairly (minimizing the maximum budget value), while trying to maximize the total delay relaxation. Other common objectives include maximum total delay relaxation under weighted, bounded, or min-skew constraints.

Figure 2 illustrates an example of these delay budget assignment policies in action. All of the nodes in the example have unit intrinsic delay, and therefore, the critical path of the graph has length 4. Delay budgets are assigned to the nodes and we assume that the timing constraint for the application is 8 time units. Hence, delay budget assignment should not create any path that takes longer than 8 units of time. For each cost function (policy), an optimal solution is depicted in the table. The additional delay budget (timing relaxation) assigned to each node is shown in each cell of the table. Note that the delay budget should be added to the unit intrinsic delay of the node to calculate its local timing constraint.

First column of the table in Figure 2, maximum budgeting, represents the result of applying this cost function. A useful extension of this policy considers weights for the nodes and tries to maximize the total weighted budget assignment. Second column of the table (bounded maximum budgeting) illustrates the node delay budgets when the cost function tries to maximize the total budget while maintaining some lower/upper bounds on the amount of delay budget assigned to nodes. In this example, node a has a lower bound of 1, and nodes b and c have an upper bound of 3 on their delay budgets. Lower/upper bounds are useful for many application domains, due to the limited local implementation choices in practice.

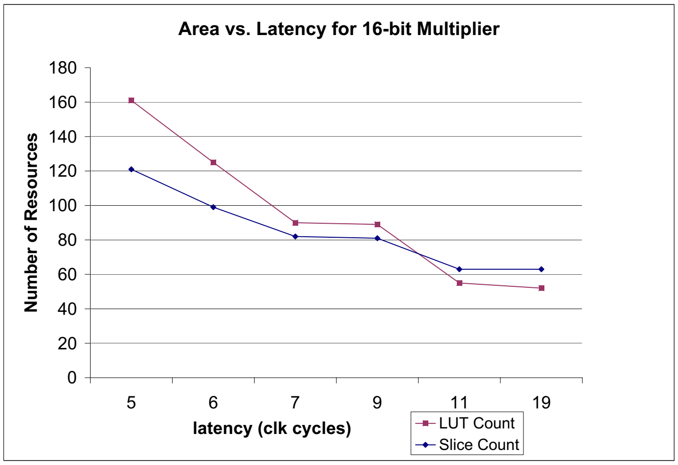


Figure 3: The relation of area and delay of 16-bit sequential multipliers of Xilinx CoreGen library

Another popular policy is to distribute the budget values fairly (minimizing the maximum budget value), while still trying to maximize the total budget. The last two columns of the table (min-max) represent the result of applying this policy to the sample graph, where 25% and 40%

degradation compared to maximum budget is allowed for each case. Note that minimizing the maximum budget value has trivial solution of zero-relaxation if there is no constraint on the total budget.

For some application domains, however, the variations in local delay constraint do not *continuously* correlate with the utility. In other words, each node might have a number of arbitrary (as opposed to consecutive) integer delay choices, each of which, correspond to some savings in the utility. In this paper, we discuss the problem of delay budgeting under arbitrary delay choices. We leverage the existing techniques to establish analytical bounds, and apply our results to V_t assignment for gate-level netlists in dual V_t technology.

3 Related Work

Time budgeting relates 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 of the solution quality. The majority of previous work have used Zero Slack Algorithm (ZSA) [31] 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. Recent results have showed that the delay budgeting problem with consecutive integer delay choices is optimally solvable in polynomial time, and have proved its superiority over ZSA through experimentation [12, 13].

In our previous work, we presented a unified theoretical framework that optimally and in polynomial time solves several widely-used formulations of integral time budgeting [33]. However, the framework is incapable of addressing the problem under arbitrarily discrete (non-consecutive integer) delay choices. The problem under arbitrary 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 showed that the solution can be approximated within any given bound (ϵ-approximation) when the application graph is a rooted tree [34].

The problem of threshold voltage assignment for leakage optimization has been known as a complex CAD problem [23]. Wang et al. [29] present an intuitive heuristic algorithm for V_t assignment. Researchers in [35] present a technique for leakage optimization via simultaneous V_t selection and circuit sizing. Khandelwal et al. focus on simultaneous V_t selection and assignment [38]. 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 theoretical limits of leakage savings using dual threshold voltages. We also significantly improve the previous results reported in [38].

From the application point of view, local delay relaxation has been utilized to improve the utility function in different applications. For example, design timing closure during different levels of VLSI CAD [22, 7, 27], energy savings via voltage and frequency adjustment [36, 3], timing-driven placement and floor planning [2, 6, 25, 24, 8], wire and gate sizing [6, 5, 18, 28], High-level synthesis [32, 20, 39], and software optimization to improve the power efficiency or computation quality of a software program [16, 34] are a few such application domains.

In addition, time budgeting closely relates to spatial budgeting, which has applications in layout compaction [42, 21, 14]. The concept of budgeting (in VLSI CAD context) was first proposed by C.K. Wong et al. [42], for spatial budgeting in layout compaction to relax the physical constraints of design components under global layout geometry constraints. The relaxation in physical constraints results in faster convergence to an improved solution.

4 Leakage And Delay Models

In this section, we briefly explain the gate leakage and delay models that are used in this paper. The leakage and delay numbers are utilized in subsequent sections to address the threshold voltage assignment problem as an implementation selection instance.

According to the BSIM model [4], leakage current of a MOS transistor can be approximated as follows:

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

where $A = \mu_0 C_{ox}(W/L)(kT/q)^2 e^{1.8}$ in which, W and L are the effective device geometries and C_{ox} is the gate oxide capacitance per unit area. The term KT/q represents the thermal voltage. μ_0 denotes the zero bias mobility and V_t is the threshold voltage. Consequently, the leakage current is exponentially dependent on threshold voltage.

Leakage current of a CMOS circuit is the sum total of the leakage currents of all of the 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 comparatively large.

In our study, we calculate the gate leakage under each input pattern. The leakage numbers under different input patterns are used to arrive at a weighted average of leakage numbers.

Subsequently, the leakage of a gate at either low or high threshold voltage is represented with one number, which is the weighted average of its leakage under various input values.

It is difficult to derive an accurate closed-form gate delay model for dual threshold voltage technology. Consecutively, we utilize the analysis and simulations performed in [38] that delivers the delay of basic gates in the library under load-dependent delay model (LDDM) [26]. In this scheme, the delay of each gate under a given threshold voltage is composed of two elements: an intrinsic delay and a slope. The gate delay is simply estimated as $intrinsic_delay + C_{load} * slope$. The delay and leakage numbers borrowed from [38] are stored in look-up tables, to enable quick estimation of circuit delay and leakage under any threshold voltage assignment.

5 Leakage Optimization Via Time Budgeting

5.1 *Problem Transformation*

We formulate the problem of gate-level V_t assignment in dual V_t technology as discrete delay budgeting [2] by which, we assign local delay constraints to gates. The delay choices for each gate are its delay under $V_{t,low}$ and $V_{t,high}$ threshold voltages considering its load. If the local delay constraint of a gate is larger than or equal to its delay under $V_{t,high}$, it will be assigned to high threshold. Otherwise, it will be assigned to $V_{t,low}$.

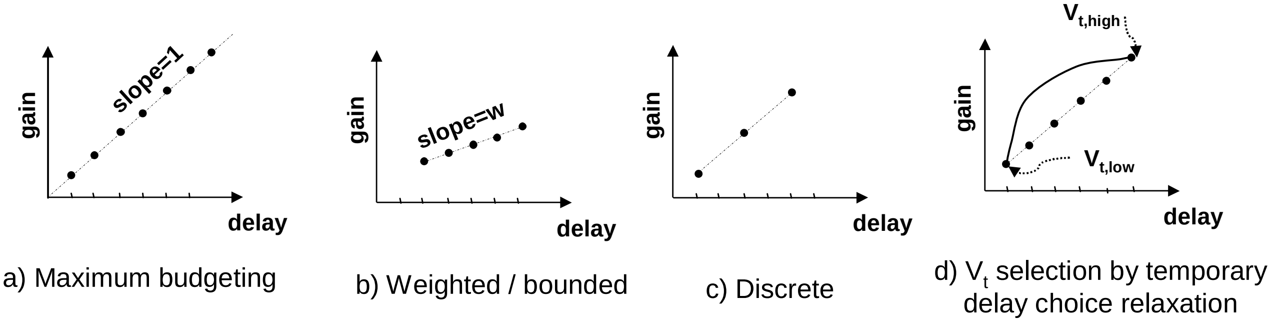
The objective of the entire process is to maximize the savings in leakage current, through assignment of selected gates to $V_{t,high}$. We might refer to this objective as the *gain* that we try to maximize. As the optimization constraint, design global timing constraint has to be met. We explain our methodology using dual threshold voltages, and later show that it is efficiently extensible to some of the cases where multiple implementations are present, including multi-threshold voltage technologies.

From a delay budgeting perspective, each gate has exactly two delay choices under dual V_t technology, i.e., there are exactly two possible points in the leakage savings-delay plane for each gate. Let us *temporarily* relax the realistic constraint of having only two discrete threshold voltages. The original two points can be assumed to form a line with a fixed slope. The slope represents the improvement of the design leakage with assignment of a unit relaxation in the delay of the gate. Note that neither 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 a very intuitive basis for assigning weights to gates (Figure 4). Weights represent relative improvement of the gain function, with unit delay relaxation at a node, and hence, they enable

[2] We use the term discrete delay budgeting to refer to case where implementations with arbitrary integer delays are available. The term integral delay budgeting refers to a special case where arbitrary integers happen to be consecutive numbers.

us to favor some gates over others when gates compete over relaxation of their local timing constraint.

The corresponding integral delay budgeting problem is to select the proper delay relaxation for each node, with a given upper bound such that the total weighted delay relaxation is maximized under global timing constraint. In the next subsection, we extend our previous results to show that this formulation is optimally solvable in polynomial time. Then, we present properties of the delay choices, under which, a budgeting instance with arbitrary-discrete delay choices can be transformed to a tractable integral budgeting instance. If a problem transformation is not possible, we round down the result of integral delay budgeting to arrive at a feasible arbitrary-discrete implementation selection solution. The proper assignment of weights guides the algorithm to distribute the timing budget among gates, so that most of them are at their either lower or upper bound of delay. Therefore, only a small amount of total gain is eliminated by rounding down the integral solution.



a) Maximum budgeting

b) Weighted / bounded

c) Discrete

d) V_t selection by temporary delay choice relaxation

Figure 4: A weighted node budgeting instance can be transformed to an equivalent weighted edge budgeting instance.

Figure 4.d visualizes different cases. As the figure suggests, the actual gain-delay variation for a gate forms a specific exponential relation. This can be seen as a special case of the arbitrary-discrete integer delay choices (Figure 4.c) with only two delay choices. However, the temporary relaxation of two delay choices converts the problem into an instance of the weighted implementation selection from a set of continuous integral delay choices with upper and/or lower bounds (Figure 4.b). We will extend our previous results [33] for solving the unit-weight unbounded version of the problem (Figure 4.a) to handle weights and bounds. This technique is then instrumented to handle discrete delay choices as well.

5.2 *Weighted Bounded Implementation Selection*

The application model that we considered so far, assumes that nodes incur delay to perform their associated computations, and edges have zero delay. More generally, we can assume that edges incur delay, and nodes have zero delay. An instance of node budgeting can be transformed to an instance of edge budgeting 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. Figure 5 depicts the idea. It follows that the problem of node budgeting is a special case of weighted edge budgeting.

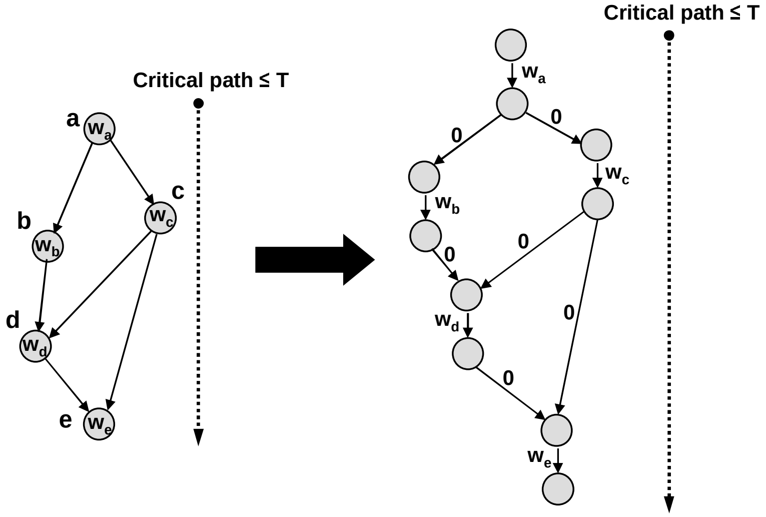


Figure 5: A weighted node budgeting instance can be transformed to an equivalent weighted edge budgeting instance.

A given node budgeting instance on a gate level netlist can be transformed according to the aforementioned three steps to form an edge budgeting instance. For simplicity, we add a virtual super input node (SI) to the netlist, that is only connected to all of the primary inputs. Similarly, we add a virtual super output node (SO) that takes only all of the primary outputs of the netlist as its fanins (Figure 6). All of the edges connecting SI or SO to any other node in G have zero delay and zero weight. Remember that an edge weight represents the relative gate leakage improvement by assigning a unit delay to the gate. The timing constraint implies that all of the paths from SI to SO must take no longer than T. The delay of each path is calculated according to the following definition.

Definition: The delay of a path p from node s to node t is equal to $\sum_{e_{ij} \in p} (d_{ij} + b_{ij})$, where d_{ij} is the initial delay of the edge, and b_{ij} is its timing relaxation (delay budget). We use the terms delay of the path, cost of the path and the distance between nodes s and t, interchangeably.

Note that any possible lower bound can be incorporated in the initial delay (d_{ij}) of the edges a priori. Hence, we assume that there is no explicit lower bound in the problem instance. The

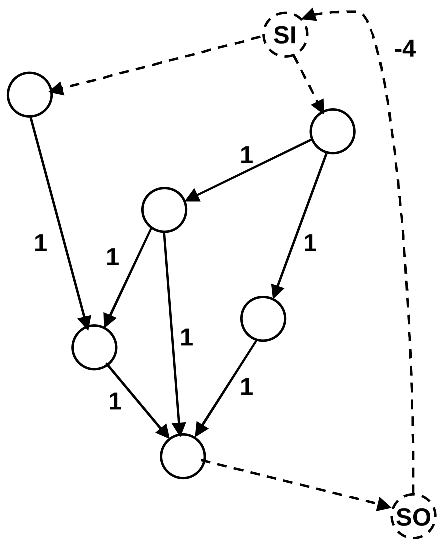


Figure 6: Sample DFG with edge delay annotations. Nodes SI, SO and additional edges are shown with dashed lines.

integral weighted, bounded (i.e., upper bounds) edge delay budgeting problem can be stated as the following ILP formulation:

$$Maximize \sum_{e_{ij} \in E} w_{ij} b_{ij} \tag{2}$$

$$\sum (d_{ij} + b_{ij}) \leq T \quad \forall SI \rightarrow SO \quad paths \tag{3}$$

$$d_{ij} + b_{ij} \leq u_{ij} \qquad \forall e_{ij} \in E \tag{4}$$

$$b_{ij}, d_{ij}, u_{ij}, T \in Z^{+} \qquad \forall e_{ij} \in E \tag{5}$$

The vectors b, d, w and u denote the delay relaxation (delay budget), initial delay including possible lower bounds, weight and upper bound for each edge, respectively. In case of threshold voltage assignment, the vector u represents the delay of gates under high threshold voltage. The following lemma is a crucial observation that allows us to re-formulate the problem.

Lemma 1 *In an optimal edge budget assignment for a given DAG, the delay of a path from a node to SO does not depend on the choice of path, and is only a function of the node.*

Proof: We prove the lemma by contradiction: Let us assume that the lemma does not hold. Hence, in the optimal solution there are some nodes whose distance to SO depend on the choice of path. Let v be the last node in a given topological ordering of the DAG, which has this property. According to our assumption, there are at least two $v \rightarrow SO$ paths that have different delays. We use the notion of $v \rightarrow u \rightarrow \ldots \rightarrow SO$ and $v \rightarrow w \rightarrow \ldots \rightarrow SO$ to represent the two $v \rightarrow SO$ paths with different delays. Note that $u \neq w$, otherwise v would not be the *last* node in the topological ordering with that property.

Without the loss of generality, we assume that the delay of the path $v \rightarrow u \rightarrow \ldots \rightarrow SO$ is smaller than the delay of path $v \rightarrow w \rightarrow \ldots \rightarrow SO$. We slow down the delay of the edge e_{vu} by unit delay. The new delay of the path $v \rightarrow u \rightarrow \ldots \rightarrow SO$ will not be greater than the delay of $v \rightarrow w \rightarrow \ldots \rightarrow SO$, because all path delays are integers. Now, we show that unit delay insertion does not violate the timing constraint of the solution.

We started with a feasible solution, which means that any $SI \rightarrow SO$ path had delay of less than or equal to T. Note that e_{vu} is the only edge whose delay is changed in the new solution. Therefore, if the timing constraint of the new solution fails, it must be due to some path going through e_{vu}. However, if the delay of some $SI \rightarrow \ldots \rightarrow v \rightarrow u \rightarrow \ldots \rightarrow SO$ is larger than T, then the delay of the path $SI \rightarrow \ldots \rightarrow v \rightarrow w \rightarrow \ldots \rightarrow SO$ is also larger than T (Remember than the delay of $v \rightarrow u \rightarrow \ldots \rightarrow SO$ is not greater than the delay of $v \rightarrow w \rightarrow \ldots \rightarrow SO$). This contradicts with the initial assumption of starting out with a feasible solution, and hence, the new solution indeed meets the timing constraint. However, this implies that the total budget of the new solution is larger than the initial solution by one, which contradicts the optimality of the initial solution. Therefore, in an optimal solution, the delay of a path from any node to SO is only a function of the node. ■

Since the delay from a node to SO is not a function of the path in the optimal solution, we can assign a variable to each node to represent this delay. Let r_i be a variable assigned to node i that represents the distance of i to SO (i to SO delay). Therefore:

$$r_i - r_j - d_{ij} = b_{ij} \quad \forall e_{ij} \in E \tag{6}$$

Utilizing this relation to rewrite the equations 2-5 leads to the following set of equations.

$$\begin{aligned}
&Max \sum_{e_{ij} \in E} w_{ij} b_{ij} && (7)\\
&r_i = r_j + d_{ij} + b_{ij} \qquad \forall e_{ij} \in E && (8)\\
&r_{SI} - r_{SO} \leq T && (9)\\
&d_{ij} + b_{ij} \leq u_{ij} \qquad \forall e_{ij} \in E && (10)\\
&r_i, b_{ij} \in Z_{+} \quad \forall v_i \in V \text{ and } e_{ij} \in E && (11)
\end{aligned}$$

Note that the number of constraints in equations 2-5 can grow exponentially with respect to the number of nodes in the graph. However, formulation 7-11 has polynomial number of constraints with respect to the problem size. Moreover, we can assume that there is a virtual edge e_{oi}, from SO to SI with $-T$ delay (Figure 6). The constraint $r_{SI} - r_{SO} \leq T$ can be represented as one of the regular edge constraints and can be safely removed as a separate constraint. The example shown in Figure 6 assumes that the timing constraint for the netlist is 4 units of time.

Utilizing equation 6, we can eliminate the budget variables (b_{ij}) from the objective function by substituting $b_{ij} = r_i - r_j - d_{ij}$. Note that the non negativity constraint of b_{ij} transforms to $\forall e_{ij} \in E : r_i - r_j \geq d_{ij}$. It follows that:

$$\sum_{e_{ij} \in E} w_{ij} b_{ij} = \sum_{e_{ij} \in E} w_{ij}(r_i - r_j - d_{ij}) =$$

$$\sum_{v_i \in V} r_i[weight_{out}(v_i) - weight_{in}(v_i)] - \sum_{e_{ij} \in E} w_{ij} d_{ij}$$

where $weight_{in}(v_i)$ and $weight_{out}(v_i)$ are the sum of weights of the incoming and outgoing edges of vertex v_i, respectively. The term $\sum_{e_{ij} \in E} w_{ij} d_{ij}$ is constant and can be eliminated from the objective function. We define $\rho_i = weight_{out}(v_i) - weight_{in}(v_i)$, which is a function of the graph structure and hence, it is constant for a given problem instance. Substituting ρ_i equations 7-11 can be rephrased as:

$$Max \sum_{i \in V} \rho_i r_i \tag{12}$$

$$r_j - r_i \leq -d_{ij} \quad \forall e_{ij} \in E \tag{13}$$

$$r_i - r_j \leq u_{ij} \quad \forall e_{ij} \in E \tag{14}$$

$$r_i \in Z_+ \quad \forall v_i \in V \tag{15}$$

The dual problem to ILP problem (12)-(15) is:

$$Min \sum_{e_{ij} \in E} u_{ij} z_{ij} - d_{ij} y_{ij} \tag{16}$$

$$\sum_{e_{ki} \in E} (y_{ki} - z_{ki}) - \sum_{e_{ij} \in E} (y_{ij} - z_{ij}) = \rho_i \quad \forall v_i \in V \tag{17}$$

$$y_{ij}, z_{ij} \in Z_+ \quad \forall e_{ij} \in E \tag{18}$$

The equations 16-18 formulate a conventional min-cost flow problem on an extended DAG, created by assigning cost $-d_{ij}$ to edge e_{ij}, and adding edge e_{ji} with cost u_{ij} to the original graph (Figure 7). y_{ij} and z_{ij} variables are the amount of flow along edges e_{ij} and e_{ji}, respectively. ρ_i is the amount of demand at node i, which is a simple function of edge weights. Equivalently, $-\rho_i$ can be interpreted as the amount of flow supply at that node. Note that the flow conservation constraint ($\sum_{i \in V} \rho_i = 0$) is satisfied as required in the min-cost flow problem [30, 11]. The dual min-cost flow problem can be solved optimally and in polynomial time, using any of the well-known min-cost flow algorithms [30]. It follows that the optimal solution to the primal problem (integral delay budgeting for edges of the graph) can be found in polynomial time [10]. Figure 7 illustrates the dual min-cost flow problem and its solution for the graph shown in Figure 6.

Once y_{ij} and z_{ij} variables (the amount of flow along edge e_{ij} and e_{ji}) are determined, we can 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 equal to the complement 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 [3] [30]. δ_i variables can be determined by utilizing any well-known shortest path algorithm, such as Bellman-Ford algorithm [37], that is applicable to graphs with negative edge costs. Figure 7 shows the residual graph and δ_i variables for the example shown in Figure 6.

Variables r_i and b_{ij} of the primal problem can be calculated by substituting $r_i = -\delta_i$ and $b_{ij} = r_i - r_j - d_{ij}$. The following theorem, which is based on the LP duality theorem and complementary slackness condition [10, 41] proves that this equation indeed determines the primal variables correctly.

Theorem 2 *$r_i = -\delta_i$ is an optimal solution to the equations 12-15, where δ_i is the shortest path of node i to SO in the residual graph.*

The time complexity of the overall algorithm is determined by the min-cost flow step, which is the slowest part of the algorithm. The technique requires the min-cost flow calculation on the augmented 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 [30], where m is the number of edges in the graph. For practical CAD problems, the degree of nodes is bound by a small constant and the graphs are sparse. Hence, the number of edges is $O(n)$ and the time complexity reduces to $O(n^2.log^2(n))$, which is quite affordable for many practical problem instances.

[3] If there is a negative cycle, the cost of the flow solution can be reduced by circulating a unit flow along the negative cycle, which contradicts the fact that we are working with the residual graph of a min-cost flow.

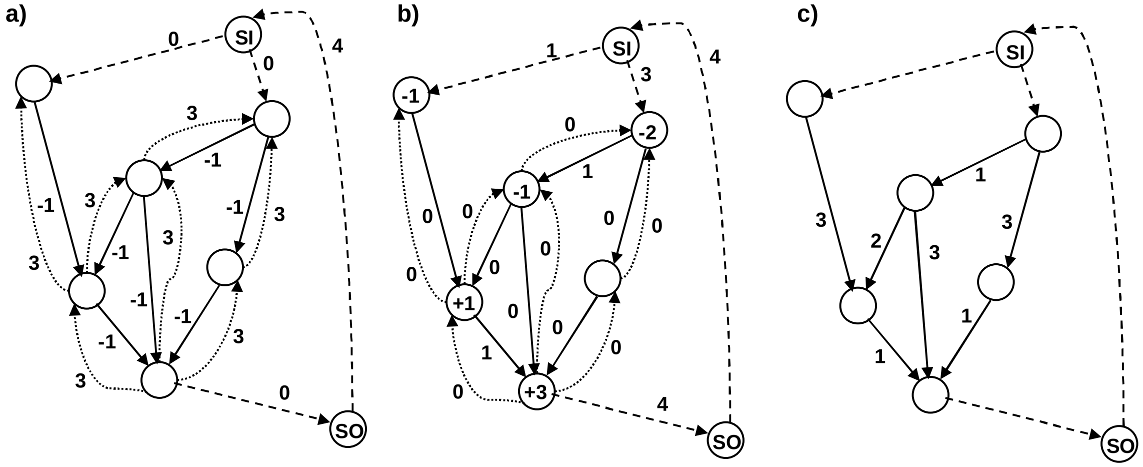


Figure 7: a)The Min-cost flow problem for example in Figure 6. Initial delays and upper bounds are represented as edge costs. b)Min-cost flow solution. Supply at each node i is $weight_{in}(i) - weight_{out}(i)$. c)Relaxed edge delays resulting in maximum total relaxation.

5.3 *Discrete Time Budgeting for Dual* V_t *Technology*

In this section, we remove the relaxation assumption of continuous integral delay choices for each gate, and focus on arbitrary-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 delay budgeting to maximize total relaxation 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 3 *The problem of discrete delay budgeting for maximization of total relaxation is NP-complete.*

Proof: We reduce the NP-Complete subset sum problem to discrete delay budgeting on a path. In the subset problem, a set of integer numbers (e_i) and an integer constant B are given. The decision problem is to find a subset of the numbers whose sum total is exactly B. The transformation procedure is as follows: For any number e_i in the given subset sum instance, we create a node with two delay choices of zero and e_i on 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 discrete delay budgeting problem on the constructed path, with timing constraint B, can be solved in polynomial time. Therefore, the discrete delay budgeting problem for arbitrary delay choices is NP-complete. ■

Although the discrete delay budgeting problem is NP-complete in the general case, a special relation between the delay choices, or the structure of the graph, might allow efficient solution of the problem. For example, under arbitrary delay choices, the optimal solution can be approximated to any given accuracy if the graph has the rooted tree structure [34]. 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 all of the nodes have delay choices that are consecutive multiples of an integer d. That is, the delay choices for node i are $m_i.d, (m_i + 1)d \ldots (m_i + n_i).d$ and the global timing constraint is T. The following lemma proves that the aforementioned instance of the problem can be transformed into continuous integral budgeting problem, and hence, can be optimally solved through the technique explained in Subsection 5.2:

Theorem 4 *The budgeting 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 budgeting 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 a consecutive-integral delay budgeting instance with lower and upper bounds of m_i and n_i for node i, respectively. The global delay constraint is reduced to $\lfloor T/d \rfloor$. Note that the original delay constraint T, 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 depend on the choice of threshold voltages. Although threshold voltages might be selected in a way that allow exact scaling, it should not generally be assumed that they follow this pattern. Therefore, we solve the threshold voltage assignment problem by relaxing it into consecutive-integral delay budgeting, solving this version, and rounding down the results to the closest feasible solution for each gate. This guarantees that we arrive at a legitimate and feasible solution. This scheme works very effectively in practice. Most of the gates with enough timing slack are assigned a timing budget equal to their budget upper bound, and hence, can be assigned to the high threshold voltage. Next section reports our observations during experiments with real circuits.

5.4 *Extension to Multiple Threshold Voltages and Other Convex Cost Functions*

In case of multiple threshold voltages, more than two implementations are available for each edge. Note that edges in the budgeting instance represent nodes (gates) in the gate level netlist (Figure 5). Figure 8 illustrates an example, where A, B, and C are three implementations of a gate with three different threshold voltages, where $V_{t,1} < V_{t,2} < V_{t,3}$. The leakage current of a gate exponentially decreases with increase of its V_t. Hence, the slopes of the line segments connecting consecutive delay-leakage points ($A - B$ and $B - C$ line segments in Figure 8) is a decreasing function of the delay, and thus $\mid w_1 \mid > \mid w_2 \mid$. The slope of each line segment reflects the relative leakage improvement per unit relaxation of the edge in the corresponding budgeting problem.

Figure 8 illustrates the required procedure to transform the three threshold version of the problem to a consecutive-integral budgeting instance. We start by temporarily relaxing the discreteness of delay choices, and assuming that all points with integral delay on $A - B$ and $B - C$ line segments are feasible solutions. We insert an intermediate node v_m on the edge connecting v_i to v_o. The $v_i \rightarrow v_m$ edge represents the $A-B$ line segment, and the $v_m \rightarrow v_o$ edge corresponds to the $B-C$ line segment of the leakage-delay curve. The forward edge $v_i \rightarrow v_m$ is annotated with the complement of the initial delay of the edge, i.e., the fastest implementation of the gate. The reverse edges $v_m \rightarrow v_i$ and $v_o \rightarrow v_m$ are annotated with the upper bound on the relaxed delay of each edge, according to Subsection 5.2. As with the dual threshold case, the integral budgeting results have to be rounded down to arrive at a feasible solution with

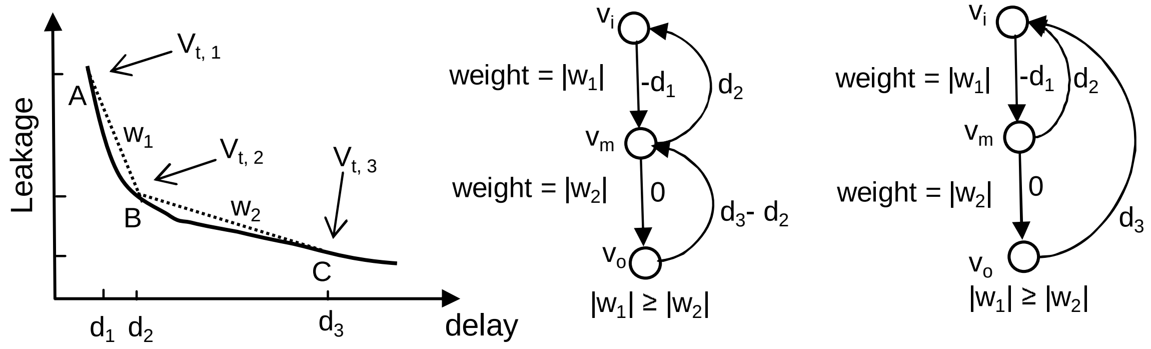


Figure 8: Handling multi threshold voltages by transformation to a budgeting instance.

The condition $w_1 > w_2$ guarantees that the edge $v_i \rightarrow v_m$ will be assigned delay relaxation, before $v_m \rightarrow v_i$. This implies that the delay relaxation will replace the implementation A with implementation B, before trying to replace it with C. The upper bound on edge $v_i \rightarrow v_m$ guarantees that after replacing implementation A with B, additional delay relaxation should be allotted to edge $v_m \rightarrow v_o$ with a smaller gain per unit delay. The transformation does not work if weight is not a decreasing function of delay. In fact, it is not hard to show that the integral version of the problem is NP-Complete, in that case.

The discussion and transformation are applicable to piece-wise linear approximation of any strictly-decreasing convex function. The leakage current of a CMOS gate as a function of its delay has this property, and therefore, our method is directly applicable. Note that a concave function, or a convex function that is increasing at some intervals violates the necessary conditions and hence, cannot be accurately modeled using our technique.

6 Experimental Results

We implemented our budgeting-based threshold assignment (BBTA) algorithm in SIS [15]. In case the temporary relaxed solution for some gate is not feasible in dual V_t technology, we round down the allocated delay to select the best feasible implementation. This is equivalent to mapping that gate to lower threshold voltage, in dual V_t technology. In other words, the temporary delay relaxation assigns any delay between delay-under-$V_{t,low}$ and delay-under-$V_{t,high}$ to a gate. If the relaxed delay is equal to delay-under-$V_{t,high}$, the gate will be assigned to $V_{t,high}$, otherwise it will be assigned to delay-under-$V_{t,low}$.

We developed our own timing analyzer and leakage estimator in which, we look up the gate leakage and delay numbers from tables that Khandelwal et al. [38] graciously provided. Their study estimates the gate leakage and delay variations with respect to V_t for gates in lib2.genlib library. In their analysis, all leakage and delay curves are normalized with respect to the basic inverter in the library.

Selected circuits from the MCNC benchmarks are mapped to the lib2.genlib library under default settings. The global timing constraint is set to the netlists critical path under $V_{t,low}$. In another set of experiments 5% relaxation is allowed to this timing constraint. 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 met the required timing constraint. Our timing analysis is based on the load dependent delay model in SIS. For each benchmark, the netlists are mapped using the built-in "map" command with no switches. The choice of benchmarks, SIS optimization, mapping switches and threshold voltages have been merely duplicated from [38] 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 show the low and high threshold voltages for each benchmark. We also repeated the same experiment with three threshold voltage whose results are reported in Table 2. Although fabrication processes with three threshold voltages might not be economical today, we performed this experiment to demonstrate the effectiveness and extensibility of our methodology, in case such processes become viable in future.

The choice of threshold voltages might be somewhat counter intuitive, since threshold voltages depend on technology rather than benchmarks. We adopted different threshold voltages for different benchmarks to be able to repeat the experiments in [38] and highlight the improvements. However, our technique is general and does not depend on the technology parameters such as threshold voltages.

Our study showed that for the choice of threshold voltages in [38], fabrication with three threshold voltages does not improve the leakage savings significantly over dual V_t technology. Intuitively, this translates to saying that two threshold voltages is the sweet spot for leakage optimization/cost metric.

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

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

The "Initial leakage" column shows the leakage of benchmarks under $V_{t,low}$ of the dual V_t technology. The next three columns report the result of applying "simultaneous V_t selection and assignment" [38], our budgeting-based threshold assignment (BBTA) algorithm with tight timing constraint, and BBTA with 5% relaxation in the timing constraint. In each case, we report the optimal results obtained by solving the corresponding MILP problem instance. Note that MILP formulations are NP-Complete in general, and finding the optimal solution by solving the MILP instance is practically inefficient for large designs. We report the optimal solutions to show the limits on power savings using dual threshold voltages. Column "Simul leakages" was calculated with 5% relaxation in the timing constraint (5% delay penalty). Hence, our algorithm results with relaxed timing constraint are contrasted against them for fair comparison. The comparison among normalized leakage current of the three methods is visualized in Figure 9. For each benchmark, initial leakage, optimized leakage after application of "simultaneous V_t selection and assignment", our budgeting-based results and the optimal MILP-based results are illustrated.

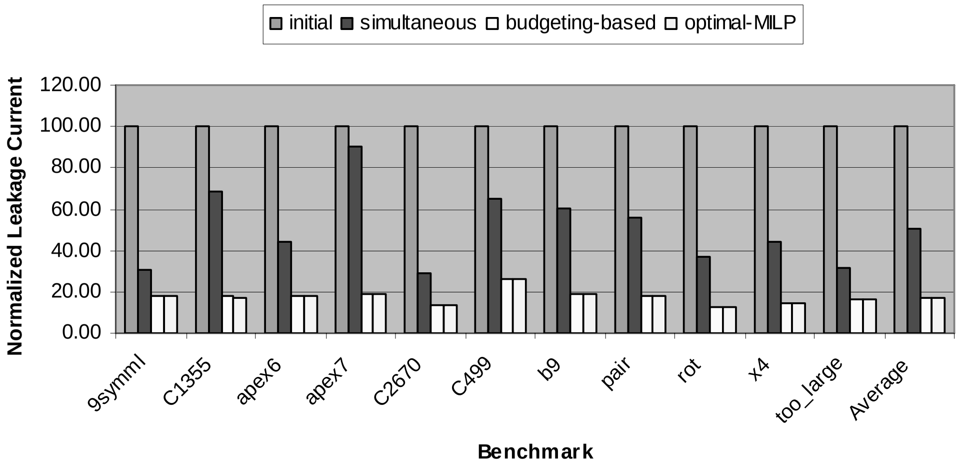


Figure 9: Comparison of normalized leakage current with 5% delay penalty.

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 reduces the leakage 49.3%, compared to the initial configuration of assigning all of the gates to $V_{t,low}$. 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. It shows that our algorithm significantly outperform Simultaneous V_t selection and assignment. More importantly, our algorithm comes very close to the limits of leakage savings using dual threshold voltages. It approaches the optimal solution, by generating results that are occasionally optimal, and on average only 0.74% worse than the optimal solution obtained by solving the MILP instances.

Circuit	V_{t_1}	V_{t_2}	V_{t_3}	Initial leakage	Simul. leakage	BBTA-tight timing			BBTA-5% relaxed timing			
						MILP	leakage	vs. init	MILP	leakage	vs. init.	vs. simul
9symml	0.40	0.46	0.50	515.7	137.54	117.3	117.8	77.16	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	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	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	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	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	666.5	667	78.22	58.19
b9	0.40	0.42	0.50	590.5	308.72	136.9	138.4	76.56	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	1261.5	1276.4	81.80	60.56
rot	0.38	0.41	0.50	4585.1	1194.89	632	632	86.22	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	423.1	423.1	85.32	59.47
too_large	0.38	0.41	0.50	2413.5	527.19	408	413	82.89	371.5	371.5	84.61	29.53
Average				3229.33	1247.18	581.74	588.32	80.30%	502.90	505.55	83.55	56.09

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

The optimality of MILP solutions comes at the price of excessive optimization effort and slow runtimes. In order to highlight the runtime advantage and practicality of our technique, we measured the runtime of our algorithm and MILP solver on the same machine. Since some of the circuits shown in Tables 1 and 2 are small, different algorithms experimented on those benchmarks run quite fast and do not allow a fair runtime comparison. In order to address this problem, we repeated the experiments on a number of large MCNC benchmarks. Table 3 illustrates the results.

The first column of the table shows the selected benchmarks. We ran the experiments with both two and three threshold voltage technologies. We assumed the values 0.4 and 0.5 volts for the low and high threshold voltages in dual V_t, and 0.4, 0.43 and 0.50 for the three possible threshold voltages in three V_t technologies. The second column illustrates the initial leakage of the circuit under lowest threshold voltage. The third column illustrates the improvements gained by applying BBTA with 5% relaxation in timing constraints, and the forth column compares the sub-optimality of our results compared to the optimal solutions obtained from MILP solver. The last column reports the runtimes in seconds measured on the same computer and relative speedup. Some columns are further divided to report the results for both two and three threshold voltage technologies. We used the GNU Linear Programming Kit (GLPK) version 4.2 to solve the MILP problem instances [17].

Circuit	Initial	improvement(%)		sub-optimality(%)		Runtime (sec)					
	leakage	$2-V_t$	$3-V_t$	$2-V_t$	$3-V_t$	MILP-2	MILP-3	BBTA-2	BBTA-3	spdup-2	spdup-3
ex5p	10037.2	85.15	85.27	0.05	0.01	10	12	0.25	0.9	40	13.33
alu4	7398.2	84.78	85.11	0.10	0.03	11	14	0.26	0.78	42.31	17.95
misex3	8470.6	85.11	85.23	0	0.01	15	17	0.25	0.88	60	19.32
seq	10558.7	85.36	85.36	0	0	18	22	0.34	1.19	52.94	18.49
apex2	10980.8	85.36	85.36	0	0	20	24	0.37	1.34	54.05	17.91
des	18737.2	85.37	85.38	0	0	26	31	0.58	1.83	44.83	16.94
spla	24127.2	85.37	85.37	0	0	119	157	1.15	5.41	103.48	29.02
ex1010	21090.7	85.33	85.36	0	0	149	187	1.49	5.24	100	35.69
pdc	31294.4	85.35	85.37	0.01	0	251	303	1.57	7.54	159.87	40.19
Average	15855.0	85.24	85.31	0.02	0.01	68.78	85.22	0.70	2.79	73.05	23.20

Table 3: Leakage Optimization and runtime comparison of optimal (MILP) and BBTA algorithms.

Our algorithm produces results that are optimal for some benchmarks, and very close to the optimal solutions for some others. On average, our algorithm is only 0.01% and 0.02% worse than the optimal solution for two and three threshold voltages, respectively. However, our solutions are obtained in considerably shorter amount of time. On average, BBTA runs 73 times faster than solving the problem instance with GLPK, while the speedup is as high as 160 for *pdc*. Essentially, the growth in size of circuits considerably slows down the solution of MILP instances, and highlights the runtime advantage of our technique. In all cases, BBTA provides solutions that are nearly optimal.

7 Conclusions

We presented the idea of time budgeting to translate a global timing constraint, to component-level local constraints. The translation is utilized to optimize some design utility such as power or area. We showed that integral delay budgeting for maximization of total delay relaxation can be efficiently and optimally solved via combinatorial techniques. The developed technique can also be used to solve weighted budgeting under lower and/or upper bounds on timing requirements.

We investigated the delay budgeting problem under discrete delay choices for components. Although NP-compete in the general case, the problem can be transformed to integral delay budgeting and effectively solved through known techniques, when certain relations between delay choices exist. We leveraged the developed theory 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%. On another set of benchmarks, our algorithm improves the leakage by more than 85% with 5% delay penalty. Compared to the optimal solutions obtained by solving the MILP instances, our algorithm is only 0.02% off, while it ran 73 times faster than GLPK MILP solver.

8 Acknowledgements

The author would like to thank professor Ankur Srivastava and Mr. Vishal Khandelwal from University of Maryland for sharing their experimental setup that made this work possible. Mr. Po-Kuan Huang of UC-Davis assisted the author in editing the paper, and his work is gratefully acknowledged. Furthermore, the author would like to acknowledge the fruitful discussions he had with professor Majid Sarrafzadeh of UCLA, and professor Elaheh Bozorgzadeh of UC-Irvine that laid the foundations of the present work [33].

References

[1] A. Davoodi, V. Khandelwal, A. Srivastava. "Variability Inspired Implementation Selection Problem". In *International Conference on Computer-Aided Design*, pages 423–427, 2004.

[2] A. Kahng, S.Mantik, and I.L. Markov. "Min-Max Placement for Large-Scale Timing Optimization". In *ACM International Symposium on Physical Design*, pages 143–148, 2002.

[3] A. Srivastava. "Simultaneous Vt selection and assignment for leakage optimization". In *International Symposium on Low Power Electronics and Design*, pages 146–151, 2003.

[4] 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.

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

[6] 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.

[7] 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.

[8] C. Yeh and M. Marek-Sadowska. "Delay Budgeting in Sequential Circuit with Application on FPGA Placement". In *ACM/IEEE Design Automation Conference*, 2003.

[9] 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.

[10] G. Dantzig. *Linear Programming and Extensions.* Princeton University Press, 1998.

[11] E. Boros, P. Hammer and R. Shamir. "A Polynomial Algorithm for Balancing Acyclic Data Flow Graphs". *IEEE Trans. Computers*, 41(11):1380–1385, 1992.

[12] 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.

[13] E. Bozorgzadeh, S. Ghiasi, A. Takahashi and M. Sarrafzadeh. ”Optimal Integer Delay Budget Assignment on Directed Acyclic Graphs”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(8):1184–1199, August 2004.

[14] E. Felt , E. Charbon , E. Malavasi and A. Sangiovanni-Vincentelli. ”An Efficient Methodology for Symbolic Compaction of Analog IC's with Multiple Symmetry Constraints”. In *Conference on European Design Automation*, November 1992.

[15] 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.

[16] 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.

[17] GNU Linear Programming Kit (GLPK). ”downloads, documentations and online manuals”. In *http://www.gnu.org/software/glpk/glpk.html.*

[18] H.R. Lin and T. Hwang. ”Power Reduction by Gate Sizing with Path-Oriented Slack Calculation”. In *IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 7–12, 1995.

[19] Xilinx Inc. ”Xilinx documentations and online manuals”. In *http://www.xilinx.com.*

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

[21] J.F. Lee and D.T. Tang. ”VLSI Layout Compaction with Grid and Mixed Constraints”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(5), September 1987.

[22] L. Singhal, E. Bozorgzadeh. ”Fast Timing Closure through Interconnect Criticality Driven Delay Relaxation”. In *International Conference on Computer-Aided Design*, pages 792–797, 2005.

[23] L. Wei, Z. Chen, M. Johnson, K. Roy, V. De. ”Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits”. In *Design Automation Conference*, pages 489–494, 1998.

[24] M. Sarrafzadeh, D. Knol and G.E. Tellez. "Unification of Budgeting and Placement". In *ACM/IEEE Design Automation Conference*, June 1997.

[25] M.Sarrafzadeh, D.A. Knol and G.E. Tellez. "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1332–1341, 1997.

[26] R. Murgai. "On The Complexity of Minimum-delay Gate Resizing Under Load-dependent Delay Model". In *International Workshop on Logic Synthesis*, 1999.

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

[28] P. Girard, C. Landrault, S. Pravossoudovitch and D. Severac. "A Gate Resizing Technique for High Reduction in Power Consumption". In *International Symposium on Low Power Electronics and Design*, pages 281–286, 1997.

[29] 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.

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

[31] 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.

[32] S. Bakshi and D. Gajski. "Component Selection for High-Performance Pipelines". *IEEE Transactions on Very Large Scale Integrated Systems*, 4(2):181–194, 1996.

[33] 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.

[34] 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.

[35] 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.

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

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

[38] 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.

[39] 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.

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

[41] L.A. Wolsey. *Integer Programming.* Wiley-Interscience Publisher, John Willey & Sons Inc., 1998.

[42] Y. Liao and C. K. Wong. "An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(2), April 1983.

Biography

Soheil Ghiasi received his B.S. from Sharif University of Technology, Tehran, Iran in 1998, and his M.S. and Ph.D. in Computer Science from University of California, Los Angeles in 2002 and 2004, respectively. He received the Harry M. Showman prize from UCLA College of Engineering in 2004. Currently, he is an assistant professor in the department of electrical and computer engineering at the University of California, Davis. His research interests include different aspects of Embedded and Reconfigurable system design.