Efficient Implementation Selection via Time Budgeting: Complexity Analysis and Leakage Optimization Case Study

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

Abstract

We present time budgeting as an efficient technique for implementation selection. We discuss discreteness in library and present an optimal algorithm for a special case of the problem. The algorithm is extended to construct a heuristic for the general case, and is experimented on the gate-level threshold voltage assignment problem in dual V_t technology. Experimental results show that our approach reduces the leakage current by close to an order of magnitude, with no or negligible delay penalty. Compared to existing algorithms, our technique outperforms a recent LP-based competitor by 33%.

1. Introduction

Implementation selection problem refers to situations, where multiple implementations of modules are available, and the designer wishes to select the proper implementation for each module to minimize the overall system cost under its utility constraint. This problem arises in many different stages of system design and VLSI-CAD. In this paper, we use time budgeting to perform implementation selection under timing constraints. We utilize time budgeting to relax the delay of the gates of a netlist, under clock period constraint. The slowed-down gates are mapped to the best available implementation in the library to improve design area, power dissipation, or other quality metrics.

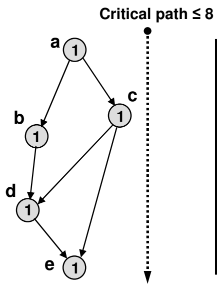
We show that discrete delay choices can make the problem NP-Complet However, under reasonable assumptions time budgeting for discrete library choices can be solved efficiently, and in some cases optimally. We apply our results to leakage optimization problem via dual threshold voltage assignment in dual 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.

2. Background

The left part of figure 1 illustrates an example of the application model used throughout the paper. We use the standard data flow graph (DFG) model, which assumes that the data flow of a given application can be represented with a directed acyclic graph (DAG). The nodes denote the computations and edges model the data dependency among nodes. Each computation starts when all its input data are available, i.e, all of its fanins have finished their computations, and takes a specific amount of time to finish its task and broadcast the result to all of its fanouts.

We assume that all primary inputs arrive at time equal to 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 modeling them as data communication nodes. 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.

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 often assumed 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, assuming that larger total budget correlates to larger improvements in the utility function. 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 [13].



	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

Fig. 1: An example of the execution model and some timing budget management policies. The table shows the optimal solution for maximum, bounded and min-max (with different total budgets) delay budget assignment. Our technique can accommodate all these policies optimally.

Figure 1 illustrates an example of different 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. For each cost function (policy), an optimal solution is depicted in the table. Note that the delay budget should be added to the unit intrinsic delay of the node to calculate its local timing constraint.

3. Related Work

The concepts of slack, operation slow down, and mobility have been extensively studied and experimented for different applications such as design timing closure [3], voltage assignment [15], timing-driven placement and floor planning, [4], gate and wire sizing [2], high-level synthesis [19], layout compaction [20] and software optimizations [8, 14]. The majority of previous work have used heuristics based on Zero Slack Algorithm (ZSA) [11] to assign timing constraints to design components. Previously, we presented optimal algorithms for integral time budgeting on a DAG [6, 13] and showed their superiority over ZSA. For discrete time budgeting, we showed that the problem is NP-Complete and the solution can be approximated within any given bound (ε-approximation) when the application graph has the form of a rooted tree [14].

4. Leakage in Deep-Submicron

We have adopted the models presented in [17] to perform our experiment on threshold voltage assignment and leakage calculation (Section 6). Therefore, we cite the relation of leakage current and delay of a MOS transistor with its threshold voltage according to their study, to maintain consistency. According to the BSIM model [1], leakage current of a MOS transistor can be approximated as follows:

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

where $A = \mu o C_{ox}(W_{eff}/L_{eff})(kT/q)^2 e^{1.8}$ and C_{ox} is the gate oxide capacitance per unit area and V_t is the threshold voltage. Consequently, the leakage current is exponentially dependent on threshold. On the other hand, delay of a MOS transistor follows the following approximate relation with respect to V_t [18, 1, 12]:

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

where α is around 1.3 for short channel and 2 for long channel devices. Leakage current of a CMOS circuit is the sum total of the leakage currents of all gates. The leakage of a CMOS gate depends on the number of transistors that are turned off and hence on the inputs. For example if a NAND gate has both NMOS transistors off (input = 00). Since these transistors are in a stack, the leakage current will be small, whereas if both PMOSs are off (input = 11) then the two off transistors are connected in parallel and the leakage is large.

5. Leakage Optimization via Time Budgeting

5.1 Transformation to Integral Weighted Time Budgeting

We formulate the problem of gate-level V_t assignment in dual V_t technology as discrete delay budgeting. The delay choices for each gate are its delay under $V_{t,low}$ and $V_{t,high}$ threshold voltages. The objective is to maximize the savings in leakage current, through assignment of selected gates to $V_{t,high}$ under a given global timing constraint. [9] reports a similar approach to address the design with dual V_t technology, however, sub-optimal heuristics are employed that lead to inferior results.

From a delay budgeting perspective, each gate has exactly two delay choices under dual V_t technology, i.e., there are exactly two points in the leakage-delay plain for each gate. Let us temporarily relax the assumption of discrete delay choices. The two points can be assumed to form a line with a fixed slope, although points in between the two ends are *invalid* choices. The slope represents the improvement of the gate leakage with assignment of a unit relaxation in the delay of the gate, and hence, serves as a very intuitive basis for assigning weights to gates.

The corresponding relaxed integral delay budgeting problem is to select the proper delay relaxation for each node, with a given upper bound, that maximizes the total weighted delay relaxation under global timing constraint. We apply our previous result [13] to solve this problem, and then round down the delay budgets to form a legal solution for the original dual-V_t-inspired discrete budgeting problem. We present properties of the delay choices, under which, a discrete budgeting instance can be directly transformed to an integral budgeting instance.

The model that we used 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. 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 2 depicts the idea.

5.2 Discrete Time Budgeting for Dual V_t Technology

In this section, we remove the relaxation 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, then 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 1. *The problem of discrete delay budgeting for maximization of total relaxation is NP-complete.*

Proof: We reduce a given instance of subset sum problem to an instance of discrete delay budgeting on a path. For each element e_i of the

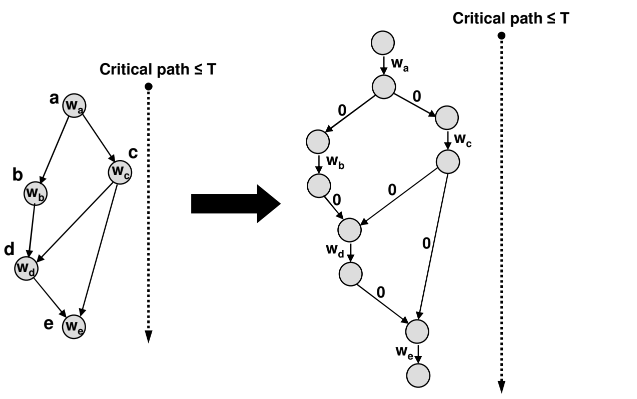


Fig. 2: A weighted node budgeting problem instance is transformed into an equivalent weighted edge budgeting instance

set, we add a node with two delay choices of zero and e_i to the path. The question "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 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 forms a rooted tree [14]. 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 are identical with delay choices that are consecutive multiples of an integer d. That is, the delay choices for each node are $m.d, (m+1)d \dots (m+n).d$ and the global timing constraint is T. The following lemma proves that such an instance of the problem can be transformed into continuous integral budgeting problem, and can be optimally solved through the technique explained in Subsection 5.1:

Theorem 2. *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.*

Proof: Omitted for brevity. ■

It follows that the aforementioned instance can be scaled down by the factor d to create an integral delay budgeting instance with lower/upper bounds of m and n, respectively. The global delay constraint is reduced to $\lfloor T/d \rfloor$. Note that the delay 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 in a way that allow exact scaling, it should not be generally assumed that they follow this pattern, especially for real circuits with different gate types. Therefore, we solve the threshold voltage assignment problem by relaxing it into integral delay budgeting, solving the integral version, and rounding down the results to the closest feasible solution for each gate. Interestingly, this scheme works very effectively in practice, and most of the gates with enough timing slack will be assigned a timing budget equal to their budget upper bound. Next section will present our experiments with real circuits.

6. Experimental Results

We implemented our algorithm for solving the integral delay budgeting problem, in SIS [7]. We perform round down on the results to convert it to a feasible discrete budgeting solution, and assign one of the two $V_{t,low}$ and $V_{t,high}$ values to each of the gates in the netlist. We used the experimental framework developed by Khandelwal et al. [17] to estimate the gate leakage and delay variations with respect to V_t for gates

Benchmark	No. of		$V_{t,high}$	Initial	Simultaneous [17]		Tight timing constraint			5% relaxed timing			runtime
	gates	$V_{t,low}$		leakage	leakage	improv. %	leakage	vs. init %	vs. simul. %	leakage	vs. init. %	vs. simul %	(seconds)
9symml	84	0.40	0.50	515.70	159.24	69.12	132.30	74.35	16.92	94.86	81.61	40.43	0.13
C1355	510	0.40	0.50	2913.40	1985.08	31.86	883.40	69.68	55.50	521.28	82.11	73.74	3.87
apex6	438	0.41	0.50	3870.40	1720.33	55.55	746.60	80.71	56.60	698.41	81.96	59.40	3.27
apex7	157	0.41	0.50	1235.10	1116.24	9.62	290.00	76.52	74.02	233.12	81.13	79.12	0.56
C2670	505	0.38	0.49	6440.30	1885.21	70.73	1022.30	84.13	45.77	865.79	86.56	54.07	6.88
C499	287	0.40	0.47	3062.60	1989.22	35.05	986.40	67.79	50.41	803.34	73.77	59.62	1.69
b9	80	0.41	0.50	590.50	357.61	39.44	142.10	75.94	60.26	110.42	81.30	69.12	0.12
pair	948	0.41	0.50	7014.71	3954.37	43.63	1387.40	80.22	64.91	1268.20	81.92	67.93	21.70
rot	403	0.39	0.50	4585.10	1692.89	63.08	686.60	85.03	59.44	596.07	87.00	64.79	3.20
x4	315	0.40	0.50	2881.30	1267.31	56.02	999.10	65.32	21.16	423.56	85.30	66.58	1.70
too_large	456	0.40	0.50	2413.50	768.16	68.17	408.50	83.07	46.82	385.38	84.03	49.83	3.70
Average				3229.33	1535.97	49.30	698.61	76.61	50.17	545.49	82.42	62.24	4.26

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

in lib2.genlib library. 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. 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 $V_{t,low}$ and $V_{t,high}$ values are the same values as the study performed by [17] to allow a fair comparison. However, 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 [17] to maintain consistency, and create a fair ground for comparison.

Table 1 summarizes our results. The first two columns report the selected benchmarks and their characteristics in terms of number of gates after mapping. Third and forth columns of the table show the value of low and high threshold voltage for each benchmark. The values are adopted from [17] to maintain consistency. The "Initial leakage" column shows the leakage of each benchmark under $V_{t,low}$. The next three columns report the result of applying "simultaneous V_t selection and assignment" [17], our algorithm with tight timing constraint, and our algorithm with 5% relaxation in the global timing constraint. The results illustrated in Column "Simultaneous" are calculated with 5% relaxation in the timing constraint (5% delay penalty). Hence, our algorithm results with relaxed timing constraint should be considered for a fair comparison. The last column reports the runtime of our algorithm in seconds measured on a PC running Linux on a 2.6 GHz Pentium4 processor with 512KB cache, and 1GB main memory.

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. With the same timing constraint, simultaneous V_t selection and assignment [17] reduces the leakage 49.3%, compared to the initial configuration of assigning all of the gates to $V_{t,low}$. If the result of simultaneous V_t selection and assignment is considered as the base for comparison, our algorithm improves the leakage 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.

7. Conclusions

We presented the idea of time budgeting to perform implementation selection under timing constraints. We investigated the delay budgeting problem under discrete delay choices for each component. 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 experiments verified the effectiveness of our approach. Our technique reduces the design leakage current by 76.6% and 82.4%, with no delay penalty and 5% relaxation in timing constraint, respectively. It outperform a novel recent technique by 33%, on average.

8. Acknowledgments

The authors would like to thank Dr. A. Srivastava and Mr. V. Khandelwal. for sharing their experimental setup, and fruitful discussions. and making this work possible.

9. References

[1] 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.

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

[3] 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.

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

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

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

[7] 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.

[8] 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.

[9] Q. Wang, S. Vrudhula. "Static power optimization of deep submicron CMOS circuits for dual VT technology". In *IEEE/ACM International Conference on Computer-Aided Design*, pages 490–496, 1998.

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

[11] 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.

[12] 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.

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

[14] 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.

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

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

[17] V. Khandelwal, A. Davoodi, A. Srivastava. "Simultaneous V_t Selection and Assignment for Leakage Optimization". *IEEE Transactions on VLSI Systems*, 13(6):762– 765, 2005.

[18] 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.

[19] 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.

[20] 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.