Transistor Level Budgeting for Power Optimization

E,Kursun S.Ghiasi M.Sarrafzadeh
{kursun,soheil,majid}cs.ucla.edu
Computer Science Department, UCLA

We present an optimal budget distribution method for low power circuit design using transistor sizing. The algorithm distributes the available budget inside the functional unit by efficient traversalof the Series Parallel Graph representation. The technique can be efficiently applied at different abstraction levels of the design as well as toward other optimization goals (such as area optimization). The complexity is O(n) in terms of the number of transistors in the circuit. Incorporating our method in the design flow yields significant improvements in power consumption. Experiments on circuits extracted from MCNC91 benchmark suite have revealed improvements up to 59% in average power and 65% in maximum power dissipation compared to an alternative budget distribution algorithm.

1. Introduction

Power consumption has become a major issue in electronics system design in the past decade. As power consumption becomes more critical, power optimization methods are incorporated at multiple levels of the design hierarchy as opposed to being restricted to a single level. Versatile techniques that can serve at various levels are highly desirable.

One such technique that can provide significant improvements at various levels of the design hierarchy is budgeting. Most of the time design decisions are made according to the critical constraints, such as worst case delay of the circuit. However, there is usually more room for optimization for the non-critical parts of the circuit without violating the constraints. Extracting and utilizing this potential relaxation on the non-critical parts of the circuit yields significant improvements in the overall design quality.Budget assignment and distribution tools are used along with other optimization methods that exploit the potential relaxation budgeting tools reveal. Among such methods are voltage scaling, transistor sizing…etc. Transistor sizing is a commonly used technique for various circuit optimization purposes. In many of the past applications, the channel widths of the transistors on the critical path are increased to improve the worst case delay of the circuit.

Given a circuit with the critical path information, the constraints on the rest of the functional units can be relaxed without affecting the worst-case delay by a budgeting algorithm. Existing tools can be used to extract potential slack of the functional units given the critical path delay. Hence, the problem reduces to distributing this given budget to individual transistors inside functional units as efficiently as possible.

In order to illustrate the importance of budget distribution scheme let us consider the example in Figure.1.

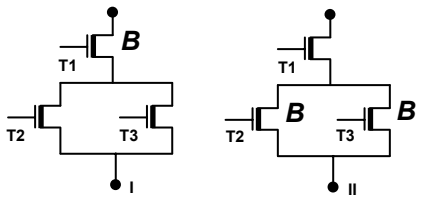

Figure 1. Alternative budget distribution schemes for example circuit.

Assume that a total size relaxation of *B%* is available to this functional unit by the initial analysis of the circuit. We refer to this value, *B*, as the budget assigned to the functional unit for the rest of the paper. The given relaxation implies that, the total amount of budget allocated to the transistors on any path from *VDD* to *GND* cannot exceed *B*. The budget can be assigned to more than one path as long as the assignment does not violate the above-mentioned constraint. Two alternative ways of distributing the given budget under these constraints is illustrated in Figure.1.

Scheme I assigns the total amount *B* to transistor *T1*. Therefore, it is unable to assign any budget to *T2* or *T3*. However, Scheme II exploits the given budget more efficiently by assigning budgets to both *T2* and *T3* of amount *B*. As a result the power improvement for Scheme II is better. As the example illustrates, the effectiveness of the budget distribution scheme plays a critical role in the quality of the overall budget allocation. In this paper, we propose a novel and optimal method to solve budget distribution on functional units in CMOS technology. This method can be incorporated into the transistor level tools to optimize the power consumption. Series-parallel (SP) graph representation is used to model the functional units. The budget distribution algorithm is then applied to allocate budget efficiently. This budget is utilized to optimize the power consumption of the functional units by transistor sizing. We illustrate that our method eases the manual effort in custom designs at transistor level as well. Yet, our budget distribution idea is applicable, but not limited to transistor level. Although we apply it for power optimization in this paper, the method is quite general and can be used toward other optimization objectives such as area minimization.

The rest of the paper is organized as follows: In Section II, we outline some related work on transistor sizing and budgeting algorithms. Then, in Section III series-parallel graph modeling of CMOS circuits is presented. In Section IV, we provide the problem

formulation. Section V presents our optimal algorithm. Experimental results and conclusions are discussed in Section VI and Section VII respectively.

2. Related Work

Transistor sizing is a commonly used technique, which is well studied in the past decades. In this Section we talk about some of the basic studies for sake of brevity. Majority of the research on transistor sizing has been on improving the critical path delay of a given circuit. TILOS [1] is a well-known example to this idea. It is a compact and strong technique with great ease of implementation. However, it does not have guaranteed convergence properties. TILOS is a heuristic approach that has been commonly used both in industry and as a building block in following academic tools. There are exact solutions to the transistor-sizing problem in this context as well. However, the running time characteristics of these approaches are not well defined [3], [4]. There are studies that target different optimization goals through transistor sizing. Sapatnekar *et al.* proposed an optimal technique for area minimization in [3]. Yet another iterative relaxation approach with two-step optimization strategy was presented by Chen *et al.* [4]. This method only deals with the cases where the delay is expressed in terms of Elmore delay model. Minflotransit [2] by Sundararajan *et al.* is a two-phase iterative relaxation based technique for area optimization. It uses a minimum-cost network flow formulation in its first phase. The transistor sizes are assumed to be fixed and the delay values are assumed to be variable at this step. In the second step the transistor delays are fixed and the sizes are treated as variable parameters. The tool also incorporates the TILOS [1] in the initial step as well, to meet the delay requirements.

Budgeting finds many application areas that vary from gate/wire sizing to VLSI layout compaction. The main goal for VLSI compaction applications is to minimize the area of the layout and budgeting idea is used to decrease the wire length [9], [10], [11]. The problem is then formulated and solved using Linear Programming (LP). However, the running time properties of LP formulation approach are not favorable. One of the most popular and efficient algorithm for delay budgeting is Zero Slack Algorithm [8] by Nair *et al*. The main application goal is to generate performance constraints for VLSI layout design. However, the algorithm is not optimal and the performance depends on the input. As a result of its popularity various improved versions of Zero Slack Algorithm have been proposed [12], [13].

Another optimal solution to the transistor sizing problem is Eins-tuner [14]. It is based on a large-scale, general purpose, non-linear optimization tool called Lancelot. The computational complexity of the tool is quite high. The running times reported in the experimental results are as high as 284,445 sec for an average sized benchmark among the benchmark set, which signifies the computational complexity of the underlying non-linear solver. Furthermore in many cases the whole non-linear optimization might not be desired when a local change is needed. These above approaches are either suboptimal or possess high algorithmic complexity. The method we propose in this work provides optimal budget distribution and *O(n)* algorithmic complexity. (*n:* number of transistors in the functional units). It is based on a different approach than many of the other complicated underlying optimization tools. It can also be used for local and global budget assignment step after the initial solution is set.

3. Series-Parallel Graph Modeling

We start by construction and basic properties of the series-parallel graphs.

Definition: The set SP of series-parallel graphs is the smallest set of graphs $G = (V, E)$ with two distinguished vertices $s, t \in V$ for which the following statements are true:

Base Graph: The graph that consists of two vertices and an edge e between them, is in *SP*. The terminals are the end points of e. Given a functional unit F consisting of transistors $\{t_1, t_2 \ldots t_n\}$ each edge e_i in SP graph represents transistor t_i in F.

Series Composition: If G_1 with terminals s_1, t_1 and G_2 with vertices s_2, t_2 are in *SP*, so is the graph G that is obtained by merging t_1 and s_2 and whose terminals are $s=s_1$ and $t=t_2$. A series composition of the SP graph represents series connection of the partitions of the circuit in our model.

Parallel Composition: If G_1 with terminals s_1, t_1 and G_2 with vertices s_2, t_2 are in *SP*, so is the graph G that is obtained by merging s_1 with t_1 to yield the terminal s and by merging s_2 with t_2 to yield the terminal t. A parallel composition in the *SP* graph represents parallel connection of the partitions in the CMOS circuits. All CMOS circuits can be modeled using SP graphs as a result of the fact that they all are built according to the above-mentioned composition statements.

4. Problem Formulation

Most of the circuit constraints are given in terms of the critical conditions such as the worst case delay of the circuit. However after the initial construction of the circuit according to the design constraints, majority of the designs can be improved in terms of various optimization goals, by utilizing the flexibility in the non-critical parts of the circuit. By making proper modifications on the non-critical parts the design quality

can be significantly enhanced, yet still be able to meet the constraints.

There are many tools, namely budget assignment and distribution tools, constructed on this basic idea. The extra flexibility in the design is extracted using an initial analysis of the circuit. This extra flexibility assigned to higher level blocks are then distributed inside the block at lower levels. Our goal in this study is to develop a budget distribution tool to efficiently distribute the provided flexibility inside the building blocks of the circuit. We assume that budget distribution takes place at an early stage and the resulting values are provided to us. These values might be generated by any budget assignment tool according to the design constraints that are not known to us. Our basic goal is to input the given flexibility provided to us, and distribute it most efficiently at the lower levels, hence improve the power consumption of the circuit. More specifically, preceding the initial analysis of the circuit a higher level budgeting tool provides potential relaxation for the individual functional units. For instance, for a functional unit *j* the delay can be relaxed by budget *B = 10%*. This corresponds to the maximum allowed delay budget for a functional unit without affecting the worst-case timing of the circuit. Delay distribution tool then finds an effective distribution of this budget among the transistors of the functional unit.

Given an SP graph *G (V, E)* and the maximum allowed timing relaxation *B*, let b_i represent the amount of budget allocated to edge $e_i \in E$. The optimal budget distribution problem then can be formulated as:

$$\max \sum_{\forall e \in E} b_i$$

Subject to: $\Sigma b_i \leq B$ (on each path in the functional unit)

We assume that the budget assigned to each edge (transistor) is directly proportional to its delay. Therefore, the formulation constraint ensures that the result will meet the design timing constraint.

5. Optimal Budget Distribution Algorithm

In this section, we present an optimal algorithm for budget distribution as formulated in the previous section. First, we mention some useful properties of SP graphs on which our budget assignment method relies. Then, we prove the optimality and efficiency of our algorithm.

Lemma.1: *Let Gain(G, T) denote the maximum total amount of budget that can be assigned to graph G, under timing relaxation T. In other words, Gain(G, T) is the maximum value of the cost function presented in section IV when B=T. Then:*

$$Gain(G, T) = (T/T') \cdot Gain(G, T').$$

Proof: We prove the lemma by way of contradiction. Assume $Gain(G, T) < (T/T') \cdot Gain(G, T')$. Let vectors **b** and **b'** denote the budget assignment distribution under timing relaxation *T* and *T'* respectively, i.e., $\boldsymbol{b}_i$ is the amount of budget assigned to edge *i*. We construct vector ***b*** by multiplying ***b'*** by *(T/T')*. Vector ***b*** created by this operation is a valid budget distribution (because it meets the constraint of having at most T unit of budget on each VCC/GND to output path) and its total budget is $(T/T') \cdot Gain(G, T')$, which is greater than $Gain(G, T)$ according to the assumption. Therefore, $Gain(G, T)$ is not the maximum budget distributed under timing relaxation *T*, which contradicts the assumption. Hence, the assumption of $Gain(G, T) < (T/T') \cdot Gain(G, T')$ does not hold. Similarly, we can construct ***b'*** from ***b*** to show that the inequality $Gain(G, T) > (T/T') \cdot Gain(G, T')$ does not hold, therefore the lemma is proved ■

Corollary.1: The budget distribution solution for an SP graph is scalable by the timing constraint.

Lemma.2: For any SP graph *G*, if $T = T_1 + T_2$, then:

$$Gain(G, T) = Gain(G, T_1) + Gain(G, T_2)$$

Proof: Let vectors **b**, $\mathbf{b_1}$ and $\mathbf{b_2}$ represent the budget distributions for *G* under timing relaxation *T*, T_1 and T_2 respectively. Assume $Gain(G, T) < Gain(G, T_1) + Gain(G, T_2)$. We construct $\mathbf{b} = \mathbf{b_1} + \mathbf{b_2}$. Therefore, the amount of budget distributed on each VCC to output (or GND to output) path by **b**, is not greater than $T = T_1 + T_2$. Hence, **b** is a valid budget distribution whose gain is equal to $Gain(G, T_1) + Gain(G, T_2)$, which is greater than $Gain(G, T)$ according to the assumption. This contradicts with the optimality of $Gain(G, T)$. Similarly, we can decompose **b** into $\mathbf{b_1}$ and $\mathbf{b_2}$ to show that $Gain(G, T) > Gain(G, T_1) + Gain(G, T_2)$ *assumption is not true, which proves the lemma* ■

Theorem.1: *Assume that SP graph G(V, E) is created by parallel composition of SP graphs* $G_1(V_1, E_1)$ *and* $G_2(V_2, E_2)$*. Then,*

$$Gain(G, T) = Gain(G_1, T) + Gain(G_2, T).$$

Proof: Let vectors **b**, $\mathbf{b_1}$ and $\mathbf{b_2}$ represent the budget distributions for *G*, G_1 and G_2 under timing constraint *T*. If $Gain(G, T) < Gain(G_1, T) + Gain(G_2, T)$, we can construct **b** by merging $\mathbf{b_1}$ and $\mathbf{b_2}$. Since, *G* is created by parallel composition of G_1 and G_2, **b** is a valid budget distribution for *G* and it meets the timing constraint of *T*. The gain of the new budget distribution for *G* is $Gain(G_1, T) + Gain(G_2, T)$, which contradicts the assumption.

If we assume $Gain(G, T) > Gain(G_1, T) + Gain(G_2, T)$, **b** can be similarly decomposed into $\mathbf{b_1}$ and $\mathbf{b_2}$ to show the a contradiction. Hence, $Gain(G, T) = Gain(G_1, T) + Gain(G_2, T)$ ■

Theorem.2: *Assume that SP graph G(V, E) is created series composition of SP graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$. Then,*

$$Gain(G, T) = MAX(Gain(G_1, T), Gain(G_2, T)).$$

Proof: Since the total budget along any path must not exceed T, the timing relaxation B has to be divided between G_1 and G_2. Assume that B_1 and B_2 represent the timing relaxation assigned to G_1 and G_2 respectively, therefore $B_1+B_2=B$.

Without loss of generality, assume that $Gain(G_1,B_1) \leq (B_1/B_2).Gain(G_2,B_2)$. According to Corollary.1, $(B_1/B_2) \cdot Gain(G_2,B_2) = Gain(G_2,B_1)$. And according to lemma 2, $Gain(G_1,B_1)+Gain(G_2, B_2) \leq Gain(G_2,B_1)+Gain(G_2,B_2) = Gain(G_2, B)$. In other words, assigning all of the timing relaxation B to one of the two constituting subgraphs will not harm the solution quality. Therefore, $Gain(G, B) = MAX(Gain(G_1, B), Gain(G_2, B))$ ■

Corollary.2: *There is an optimal budget assignment in which, edges receive either budget B or no budget at all.*

In summary, we proved that for optimally assigning the budget to an SP graph, we should solve two subproblems of smaller size at each parallel composition, and one problem at each series composition point. The subproblems can be also solved recursively according to the same rule. However, the question of "which subgraph to pick at a series composition point" is not answered yet. We will address this issue in the rest of this section. The aforementioned method optimally assigns the budget B to some of the edges in the graph. However, it does not try to distribute the budget to many edges of the graph. The optimal budget assignment can be distributed according to the following lemma:

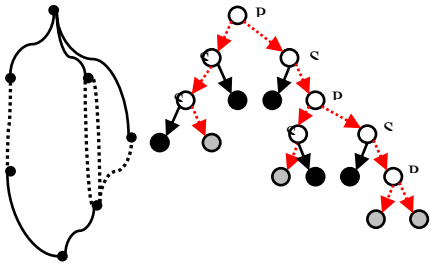

Figure 3. An SP graph and its corresponding binary tree. Leaves denote graph edges and internal nodes represent composition rules. The budget is assigned to gray leaves, which are denoted by dashed edges in graph.

Lemma.3: Given a path of k edges with budget B assigned to one of them, the budget can be reassigned such that all k edges are relaxed by budget B/k. This budget reassignment preserves the solution's quality (optimality).

The results of Lemma 3 can be integrated with Corollary 2 to construct the optimal budget assignment algorithm. By definition, an SP graph *G (V, E)* can be modeled using a binary tree, whose internal nodes represent series (S) and parallel (P) composition rules and its leaves denote edges with two terminals (see Figure.3). According to Corollary.2, there is an optimal solution in which, each tree leaf is either selected for assigning budget B or not selected at all. It follows that the optimal algorithm can traverse the tree in a bottom-up fashion to select the proper leaves. It can make decisions as to what path to take at "S" nodes, based on the number of leaves that can be relaxed by budget B. The information of such leaves can be propagated to upper levels recursively. Figure 3 illustrates a sample SP graph and the corresponding optimal solution.

Furthermore, according to Lemma 3 the budget assigned to an edge can be safely reassigned among other adjacent edges on the same serial path. Note that nodes on the serial path can be connected to other parts of the graph only at either terminals of the path. For finding serial paths of an SP graph, immediate nodes with type 'S' are merged into one node. Then, all of the children of the new 'S' super-node are on the same serial path and the budget can be reassigned among those that are leaves, while preserving the optimality of the solution.

Figure.4 outlines the pseudo code of the optimal budget distribution algorithm. *"relaxable edges"* in the pseudo code represent those edges of the graph that can potentially accept the budget B. The maximum budget that can be assigned is determined at the root. The edges that contribute to the maximum budget should be relaxed by B/k, where k is the number of edges in their corresponding serial path.

```
Algorithm OPTIMAL-SP-Budgeting:
For all nodes of the tree in reverse order of their level Do:
    if node is a leaf
        relaxable edges = 1;
    elseif node is a "P" operation
        relaxable edges = relaxable edges of left child +
                          relaxable edges of right child;
    else ( node is a "S" operation)
        relaxable edges = MAX (relaxable edges of left child,
                               relaxable edges of right child)
maximum achievable budget = root->relaxable edges;
for all edges that contributed to root->relaxable edges
        Budget (e_i) = B / # of edges in the corresponding serial path;
```

5.1. Complexity Analysis

Algorithm OPTIMAL-SP-Budgeting, visits each node of the binary tree exactly once. Hence, it runs in *O(n)* time in terms of the size of the input SP graph. It is evident that this also constitutes the lower bound for any other algorithm, because any algorithm has to read the input graph, which requires linear number of operations in terms of the size of the input. Therefore, our algorithm is optimal in terms of both complexity and solution quality.

6. Experimental Results

We used functional units extracted from MCNC91 benchmarks for our experimental analysis. The input benchmark expressions from MCNC91 combinatorial suite were optimized using SIS [7] Algebraic Optimization Script. The resulting optimized logic-level functions were then transferred to transistor level circuit layouts and inputted to HSPICE. The simulations were performed at *0.18µm* process technology. The functional units vary from 20 transistors to as large as 62 transistors. HSPICE is used for average and maximum power consumption analysis on the generated circuits. Randomly generated input vectors are used to drive the input voltages for the simulations. Figure.6 demonstrates the change in *average* power consumption over the given budget values. The *x-axis* indicates different budget values for the experiments that range from *10%* to *300%*.

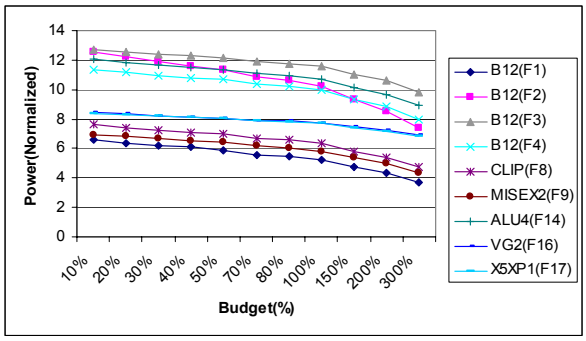

Figure 6. Average Power Consumption of benchmark circuits over different budget values (power values are normalized)

The budget values are selected from a wide range to encompass the range that can be assigned to the functional units by the initial budgeting algorithm. It is important to note that functional units are assigned different budget values according to the criticality of the delay that they impose on the circuit. Those functional units that are contributing to the high-delay paths (but not on the critical path) might be assigned budgets as low as *10%* or even less. However, for functional units that are not on the critical path, much higher budgets can be assigned. The decrease in maximum power consumption follows a similar pattern with the average consumption as well. It is important to note that the results in power gain vary for different algorithms. In order to illustrate the effects of budget distribution algorithm on the overall results, we have conducted a second set of experiments with an alternative budget distribution algorithm. This algorithm is based on a similar idea to commonly used Zero Slack Algorithm. The amount of budget is distributed to the transistors without special preference on any transistor and the scheme tries to maximize the number of transistors that are affected by the budgeting.

Table 2 and 3 illustrate the percentage difference in the power consumption results between the two algorithms. As the results indicate, the amount of gain that can be attained through budgeting is heavily dependent on the budget distribution method. Our algorithm outperforms in all of the benchmark circuits for all budget values. The difference in the resulting power consumption is as high as *59%* and *65%* in average and maximum power dissipation values respectively. In general the running time was in the order of seconds for the experimented functional units. The complexity of the algorithm is optimal in terms of the input size, which is linear in number of transistors in the circuits.

7. Conclusion

In this paper we propose an optimal and efficient budget distribution technique at transistor level. Our experiments illustrate that this method yields significant power improvements. The improvements are as high as *59%* in average and *65%* in maximum power consumption compared to an alternative budget distribution algorithm based on a similar idea to ZSA. Furthermore the algorithm has linear time complexity, which is optimal in input size. In this work we applied our budget distribution idea to transistor level power optimization. However, our technique is general and can be applied toward other optimization objectives.

8. References

[1] J. P. Fishburn and A. E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing", Proceedings of the 1985 International Conference on Computer-Aided Design, pp. 326-328, November 1985.

[2] V. Sundararajan, S.S. Sapatnekar, K.K. Parhi, "MINFLOTRANSIT: Min-Cost Flow Based Transistor Sizing Tool", Design Automation Conference, 2000

[3] S. Sapatnekar, V. Rao, P. Vaidya, S. Kang, "An exact solution to the Transistor Sizing Problem for CMOS Circuits using convex optimization", IEEE Transactions on Computer Aided Design, vol. 12, pp. 1621-1634, 1993.

Table 2. Percentage difference between maximum power dissipation of Optimal-SP-Budgeting and alternative method

% Diff (Avg)	10%	20%	30%	40%	50%	70%	80%	100%	150%	200%	300%
B12(F1)	2.645	5.229	7.706	8.099	12.301	17.237	19.212	22.433	33.231	42.437	59.069
B12(F2)	2.379	4.498	6.706	8.789	10.917	14.922	16.78	20.006	28.703	36.874	50.78
B12(F3)	1.072	2.096	3.127	4.146	5.162	7.1513	8.0879	9..962	14.48	19.123	27.222
B12(F4)	1.464	2.895	4.189	5.51	6.7404	9.2557	10.381	12.989	18.531	23.872	33.792
CLIP(F8)	2.295	4.566	6.724	8.847	11.074	15.193	17.388	21.342	31.005	40.646	58.09
MISEX2(F9)	1.959	3.956	5.558	7.327	9.092	12.354	14.019	17.221	24.606	31.05	44.424
ALU4(F14)	1.453	2.849	4.485	5.57	6.799	9.288	10.492	12.863	18.279	23.126	31.526
VG2(F16)	1.085	2.112	3.068	4.043	4.986	6.644	7.4311	8.9302	12.217	14.908	19.062
X5XP1(F17)	1.1	2.149	3.092	4.068	4.985	6.658	7.4658	8.984	12.242	14.976	19.17

Table 3. Percentage difference between average power dissipation of Optimal-SP-Budgeting and the alternative

% Diff (Max)	10%	20%	30%	40%	50%	70%	80%	100%	150%	200%	300%
B12(F1)	3.015	5.827	8.647	8.89	13.945	19.694	22.081	26.651	37.87	47.864	65.608
B12(F2)	2.469	4.753	7.05	9.197	11.365	15.516	17.406	21.238	29.887	37.52	50.166
B12(F3)	1.777	3.39	5.112	6.783	8.305	11.448	12.859	15.649	22.712	30.465	40.326
B12(F4)	1.823	3.609	5.231	6.885	8.500	11.554	12.975	15.861	22.484	28.649	39.166
CLIP(F8)	1.186	2.411	3.468	4.535	5.655	7.3169	8.882	10.676	15.267	20.217	28.685
MISEX2(F9)	1.262	2.578	3.772	4.923	6.168	8.3118	9.469	11.602	16.588	21.202	29.388
ALU4(F14)	1.878	3.766	6.151	7.308	8.997	12.361	13.986	17.182	24.664	31.626	44.298
VG2(F16)	2.677	5.234	7.65	10.06	12.499	17.046	19.204	23.517	33.347	41.894	57.678
X5XP1(F17)	2.463	4.854	7.089	9.355	11.559	15.692	17.685	21.569	30.335	38.198	51.719

[4] C. Chen, C.N. Chu, D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation", Proceedings of 1998 IEEE/ACM International Conference on Computer Aided Design, pp. 617-624, 1998.

[5] H.Y. Chen, S. M. Kang, "A Circuit Optimization Aid for CMOS High Performance Circuits", Integration VLSI Journal, Vol. 10, pp. 185-212, 1991.

[6] Z. Dai and K. Asada, "MOSIZ: A two-step Transistor Sizing Algorithm Based on Optimal Timing Assignment Method for Multistage complex Gates", Proceedings of the 1989 Custom Integrated Circuits Conference", pp. 17.3.1-17.3.4, 1989.

[7] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", Technical Report No. UCB/ERL M92/41, University of California Berkeley, 1992.

[8] R. Nair, C.L. Berman, P.S. Hauge, E.J. Yoffa, "Generation of Performance Constraints for Layout", Proceedings of IEEE Transactions on Computer Aided Design, Vol.8, No.8, pp 860-874, August 1989.

[9] Y. Liao, C.K. Wong, "An Algorithm to Compact VLSI Symbolic Layout with Mixed Constraints", Proceedings of IEEE Transactions on Computer Aided Design, Vol.2, No.2, April, 1983.

[10] J.F. Lee, D.T. Trang, "VLSI Layout Compaction with Grid and Mixed Constraints", Proceedings of IEEE Transactions on Computer Aided Design, Vol.6, No.5, Sep, 1987.

[11] E. Felt, E. Charbon, E. Malavasi, A. Sangiovanni-Vincentelli, "An Efficient Methodology for Symbolic Compaction of Analog ICs with Multiple Symmetry Constraints", Proceedings of Conference on European Design Automation, Nov., 1992.

[12] T. Vao, P.M. Vaidya, C.L. Liu, "A New Performance Driven Placement Algorithm", Proceedings of International Conference on Computer Aided Design, pp.4447, 1991.

[13] H. Youssef, E. Shragowitz, "Timing Constraints for Correct Performance", Proceedings of International Conference on Computer Aided Design, pp.2427,1990

[14] A.R. Conn, I.M. Elfadel, W.W. Molzen, P.R. O'Brien, P.N.Strenski, C. Visweswariah, C.B. Whan, "Gradient-Based Optimization of Custom Circuits Using a Static-Timing Formulation" Design Automation Conference 1999.