Clock Buffer and Wire Sizing Using Sequential Programming

Matthew R. Guthaus*
Department of EECS
University of Michigan
Ann Arbor, MI
mguthaus@eecs.umich.edu

Dennis Sylvester
Department of EECS
University of Michigan
Ann Arbor, MI
dennis@eecs.umich.edu

Richard B. Brown
Department of ECE
University of Utah
Salt Lake City, UT
brown@coe.utah.edu

ABSTRACT

This paper investigates methods for clock skew minimization using buffer and wire sizing. First, a technique that significantly improves solution quality and stability of sequential programming-based buffer/wire sizing is used. Then, a new formulation of clock skew minimization that uses quadratic programming and considers sub-critical skews in addition to the most critical skews is presented. The quality of results are verified to be more robust using Monte Carlo simulations to account for process sensitivity. For the same power budget, the sequential quadratic programming (SQP) method has better expected skew, standard deviation, and overall CPU time on average.

Categories and Subject Descriptors
J.6 [Computer-Aided Engineering]: Computer-Aided Design

General Terms
Algorithms, Design.

Keywords
Clock tree synthesis, skew, robust design.

1. INTRODUCTION

Clock distribution is vital to all synchronous integrated circuits; a poor clock distribution network can result in limited speed, high power consumption, and non-functional circuits. Past research on clock synthesis has examined clock skew scheduling [9], clock routing [4, 6, 14], wire sizing [14, 18], buffer insertion and sizing [2, 14, 19, 21], and process sensitivity analysis [1, 10, 15]. However, no works have looked at the efficiency and quality of algorithms for tuning routed, buffered clock trees and the impact that these algorithms have on parametric yield. In this paper, the initial clock trees are buffered and routed according to prior methods [4, 19] and improved through wire and buffer sizing only. Several previous works have examined clock skew scheduling and the impact on process variation [13, 16], but these do not consider the construction of the actual clock tree and do not perform analysis with multiple inter- and intra-die sources of variation.

The objective of this paper is to minimize skew given a power budget. This is useful in the final stages of a design when the clock loads have changed, useful skew is being leveraged, or more accurate parasitics, such as coupling information, are available. The methods that are proposed are independent of the buffer and interconnect models unlike previous algorithms [7, 12, 20] and can consider the effects of slew rates on buffer delay and resistive shielding in interconnect delay.

The local skew between any two sinks i and j is $|d_i - d_j|$ where d_i (d_j) is the arrival time of sink i (j). The maximum skew, or just "skew", of a clock is defined as the maximum difference between any logically connected pair of sink arrival times as in

$$\text{Skew} = \max_{i,j \in Sinks} (d_i - d_j) \quad (1)$$

$$= \max_{i \in Sinks} (d_i) - \min_{j \in Sinks} (d_j). \quad (2)$$

If the underlying logic is unknown, the skew between all pairs implies the simpler, equivalent second statement. In general, each sink delay, d_i or d_j, is a nonlinear function of the buffer and wire sizes. Therefore, clock skew optimization is a non-linear problem.

Reference [21] proposed clock tree buffer sizing for power minimization using sequential linear programming (SLP). However, there are two significant drawbacks with the proposed SLP formulation. First, the linearity of the subproblem is assumed valid in a user-defined constant range. A static value results in suboptimal solutions and increased run-times. Second, minimizing the maximum skew of a clock tree can produce trees that are susceptible to process variation induced skew. If many sink pairs have skews equal to the maximum skew and have differing sensitivities to the sources of variation, it is more likely that each chip will be manufactured with a poorly skewed clock tree. Both of these concerns are addressed in this paper.

In Section 2, we give a brief introduction to the method of sequential programming (SP) for clock tuning. In Section 3, we present a heuristic feedback technique called an adaptive confidence range that dynamically changes the valid range of the subproblem. In Section 4, we present a modified clock skew problem that uses a quadratic objective function and

*M. R. Guthaus is now an Assistant Professor of Computer Engineering at University of California Santa Cruz.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

sequential quadratic programming (SQP). In Section 5 and 6, we describe the sources of variation that are modeled in this work and compare the new quadratic programming approach to existing algorithms. Lastly, we make conclusions and propose future work in Section 7.

2. SEQUENTIAL PROGRAMMING

Sequential programming (SP) is a well known method to optimize non-linear problems [17]. The outline of a simple Sequential Linear Programming (SLP) algorithm is shown in Algorithm 1. The general algorithm iteratively improves an objective function by solving a sequence of simplified subproblems. Each subproblem at iteration i, P^i, is a linear (or quadratic) approximation of the non-linear problem and is less difficult to solve than the non-linear problem itself. The exponent specifies the associated iteration of a solution, subproblem, or gradient. The subproblems can be solved using known mathematical techniques such as simplex, conjugate gradient, or successive over-relaxation. The solution resulting from the subproblem of an iteration is hopefully a better solution of the non-linear problem than the previous solution, but this cannot be guaranteed. SLP methods can have poor convergence on some non-linear problems.

Algorithm 1 Sequential Linear Programming algorithm.

1: $i \Leftarrow$ Set iteration to 0
2: $x^0 \Leftarrow$ Initial solution
3: **repeat**
4: $\quad G^i \Leftarrow$ Calculate gradient
5: $\quad P^i \Leftarrow$ Setup subproblem(G^i)
6: $\quad x^{i+1} \Leftarrow$ Solve subproblem(P^i, x^i)
7: $\quad i \Leftarrow$ Increment iteration, $i + 1$
8: **until** convergence test satisfied **or** max iterations
9: Return x^i

This work formulates clock tree buffer and wire sizing as skew minimization given a power budget. The remainder of the paper uses the following notation:

- x is a vector of buffer/wire sizes, x_i.
- Δ is a vector of buffer/wire size changes, δ_i.
- D is a vector of present sink delays, d_i.
- G is the matrix of the first-order sensitivity, $\frac{\partial d_i}{\partial x_j}$, of each sink delay, d_i, with respect to each buffer/wire size, x_j.
- S is a vector of the new sink delays, s_i.
- s_{max} (s_{min}) is the new slow (fast) sink delay.
- P_{max} (S_{max}) is the power (skew) bound.
- P_{cur} is the present power consumption.
- β is a vector of power sensitivities per buffer/wire size change.

For SLP clock buffer/wire sizing, the subproblem of Algorithm 1 is the following linear program:

$$\begin{aligned} \min \quad & s_{max} - s_{min} && (3)\\ \text{s.t.} \quad & s_i - s_{min} \geq 0, \quad \forall i \in \text{Sinks} && (4)\\ & s_{max} - s_i \geq 0, \quad \forall i \in \text{Sinks} && (5)\\ & D + G\Delta = S && (6)\\ & P_{cur} + \beta\Delta \leq P_{max} && (7) \end{aligned}$$

The maximum skew objective, (3), is adapted from the definition of skew, (1). All skew constraints could be explicitly listed, but since this paper considers the skew between all pairs of sinks, the simplification in (2) is used to reduce the number of skew constraints from quadratic (n^2 for each pair of n sinks) to linear ($2n$ for a max/min constraint on each of n sinks) as can be seen in (4)-(5). Each sink is associated with a delay equality constraint that describes the delay in terms of the present delay and all buffer/wire size changes. This equality constraint can be written as,

$$s_i = d_i + \sum_{\forall j} \frac{\partial d_i}{\partial x_j}\delta_j,$$

for each sink i and buffer/wire j. All such constraints can be written in matrix form using the gradient (G), the vector of size changes (Δ), and the present and new sink delays (D and S, respectively) in (6). The optional power constraint uses the power sensitivity of each buffer/wire size along with the present power consumption and the buffer/wire size changes to limit the solution power consumption. Since the switching frequency and voltage do not change, the power consumption is approximately proportional to the solution capacitance.

In our implementation of SP for buffer/wire sizing, the gradient, G, is calculated using finite differencing with incremental timing analysis. A single timing analysis calculates the present delay of each sink, d_i, with no size changes. The gradient computation then iterates through each buffer and wire size and approximates the gradient with

$$\frac{\partial d_i}{\partial x_j} \approx \frac{d_i(x_j + \epsilon) - d_i(x_j)}{\epsilon} \qquad (8)$$

where ϵ is the amount of the perturbation and $d_i()$ is the sink delay as a function of the wire/buffer size. All sink sensitivities to a particular variable can be found in a single evaluation.

A power minimization problem can be made by replacing the objective function with

$$\min \qquad P_{cur} + \beta\Delta$$

and the power constraint with a skew constraint

$$s_{max} - s_{min} \leq S_{max}.$$

This work, however, investigates the problem as power-constrained skew minimization rather than skew-constrained power minimization. The power-constrained formulation has the advantage that an initial feasible solution is trivial: size every buffer/wire to the minimum. In the skew-constrained formulation, such a simple feasible solution is not possible and a constraint penalty method must be used at the expense of the overall run-time.

For SLP to be successful, constraints are added to the previous linear program subproblem, (3)-(7), to limit the size change of each buffer and wire during an iteration. This limit specifies a small region where the gradient's linear approximation is valid and can be formalized with

$$\max(L_i, x_i - \epsilon_i) \leq x_i + \delta_i \leq \min(U_i, x_i + \epsilon_i) \qquad (9)$$

where x_i is the present size and δ_i is the size adjustment. L_i and U_i are hard lower and upper limits, respectively, specified by the library and process technology.

3. ADAPTIVE CONFIDENCE RANGE

The ϵ_i parameter in (9) defines a range for each variable where the linear subproblem is assumed valid. If it is very

small, many iterations will be needed to converge on a good solution. In addition, if the skew can not be improved significantly in the range, the algorithm may exit thinking it has found an optimal solution. If ϵ_i is too big, the gradient may not be valid over the whole range and the resulting solution of the subproblem could worsen the tree. The algorithm would then exit prematurely.

This section proposes an enhancement to the SLP method that dynamically changes the ϵ_i values based on feedback from the previous iteration. The feedback heuristic is similar in spirit to more formal trust region methods, but requires significantly less computation [17]. Since clock tree buffer and wire tuning is better behaved than general purpose non-linear optimization, this simple heuristic provides ample guidance in finding a good solution.

We define a confidence factor, η, that is the percentage of the permissible size range of a buffer or wire. This confidence factor defines the allowable range as

$$\epsilon_i = \eta \times (U_i - L_i). \tag{10}$$

Since η is unit-less, the same value can apply to both buffer and wire sizes for each of the variables that are constrained by (9) and (10). Abstractly, η specifies an n-dimensional hypercube as the region of validity for the subproblem. More traditional trust region methods use per-variable regions of validity and spend considerable effort computing this region. As the later results will show, a single value of η is sufficient for the purposes of buffer and wire tuning in clock trees.

The value of η is dynamically updated during each iteration to find the largest region in which the subproblem is a good approximation. After the subproblem is solved in each iteration, the correlation between the linear approximation and the actual clock tree skew is evaluated using the following metric

$$\rho = \frac{\text{Est}_{i+1} - \text{Est}_i}{\text{Act}_{i+1} - \text{Act}_i} \tag{11}$$

where Est_i and Est_{i+1} are the linearly estimated skew value before and after the subproblem is solved, respectively and Act_i and Act_{i+1} are the actual skew before and after the solution to the subproblem is applied to the clock tree, respectively. ρ is not the statistical correlation coefficient; however, it does represent how well the subproblem's estimated skew and the actual skew correlate. The numerator of ρ is always non-negative since the initial subproblem is feasible and, in the worst case, cannot be improved. The denominator can be positive or negative. If ρ is negative, our improvement actually worsened the solution. The ideal value for ρ is 1 since this means that our approximated skew perfectly predicted the actual skew. A value larger than 1 means that the objective under-predicted the skew and a value smaller than 1 means that the objective over-predicted the skew. This leads to a heuristic schedule for updating the confidence value,

$$\eta^{i+1} = \begin{cases} \frac{1}{4}\eta^i, & \text{if } \rho \leq 0 \\ 2\eta^i, & \text{if } \rho \geq \frac{3}{4} \\ \eta^i, & \text{otherwise} \end{cases}$$

In the first case, the subproblem solution did not correctly predict the actual result so the confidence factor is decreased and the new solution is rejected. In the second case, our subproblem correctly predicted improvement, so the present solution is kept and the confidence factor is increased for the next iteration. Otherwise, the objective function was a reasonable approximation of the actual skew so the solution is kept and no adjustments are made to the confidence factor. Efficient reuse of the gradient and incremental timing analysis for calculating the actual skew reduces the overhead of each unsuccessful iteration.

SLP buffer and wire sizing of clock tree benchmarks was performed with and without the adaptive confidence range to study the impact on both solution quality and CPU time. A wide range of static values for η were compared to the adaptive confidence method. The results for one benchmark, r1, are typical and are shown in Table 1. In all benchmarks, the final skew was better and the number of iterations until convergence was significantly reduced by using the adaptive technique. As previously described, a large confidence factor prevents convergence to a good solution whereas a small confidence factor results in exorbitant run-time. It is important to note that a smaller confidence factor does not always mean a better solution. In the case of 1.5% confidence, the final skew was actually better than if the confidence factor was 0.5%. This is due to the non-linear nature of the problem and the fact that a small range may not provide enough improvement for the SP algorithm to continue optimization. All further comparisons use the adaptive confidence technique.

Table 1: The effect of a static and adaptive confidence factor.

Confidence % (100η)	Skew (ps)	CPU (ps)	Iter.
25.00	56.5	6.2	1
10.0	51.9	9.3	2
2.5	49.1	22.0	7
1.5	47.6	45.7	16
0.5	48.7	94.1	35
0.25	48.4	197.9	74
Adaptive	46.3	35.1	18

4. QUADRATIC OBJECTIVE

A large number of equally critical paths in a design can result in circuits with lower parametric yield due to process variations. Having a large number of nearly critical paths means that it is more likely that one of these paths will be critical in a manufactured die. This idea has been called the "slack wall" for data-paths [3]. In clock tree skew optimization, a similar skew wall forms. For example, consider the sink skew histogram shown in Figure 1. This wall of nearly critical skews makes it more likely that when a chip is manufactured, one of the skew pairs will be worse than expected due to process parameter variations. The actual worst skew pair may not be the worst identified by deterministic static timing analysis. Considering the skew wall during optimization allows an algorithm to trade-off improvements in many skew pairs versus the single most critical pair and can increase the overall performance and robustness of a design.

While there have been significant improvements in statistical timing analysis during recent years, there are still inadequate models of variability and correlation for direct statistical optimization. In addition, it is impractical to use general purpose Monte Carlo simulations in the inner loop of optimization. Bai et al. [3] proposed to penalize

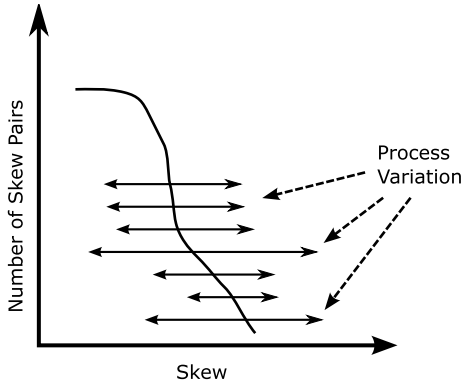

Figure 1: Skew histogram.

the objective function for gate sizing with an exponential term so that nearly equal critical paths are avoided. This section proposes an alternative method that considers sub-critical skews in the objective function using a quadratic cost function. Minimizing the total skew squared instead of maximum skew improves the distribution of critical skews so that a skew wall is avoided. It is emphasized that we are not approximating maximum skew using a second-order function as typically done in sequential quadratic programming (SQP). The objective of the SP subproblem is actually changed to a quadratic cost to shape the skew wall such that it results in a more robust solution. In doing this, improvements to CPU time are also observed.

Several interesting properties can be leveraged when using a quadratic model for clock skew. The most obvious property is that the quadratic objective is a continuous function whereas maximum skew can be non-differentiable. In addition, methods of efficiently solving large scale quadratic programming problems have been well studied. When considering sub-critical skews, the objective can be improved by reducing the skew of the worst pair or by reducing the skew of several sub-critical pairs. The quadratic nature ensures that the worst skew is not ignored since it contributes increasingly more to the objective than sub-critical skews. As an analogy, the quadratic cost can be thought of as connecting each pair of sinks with a spring in the time domain and then trying to minimize the total energy in the system of springs. If the data-paths between certain pairs of sinks are more critical than others, the spring constant can be increased to reflect this criticality. The general formula of our new objective is

$$\Phi(S) \quad = \quad \sum_{i>j} w_{ij}(s_i + b_{ij} - s_j)^2 \tag{12}$$

where w_{ij} is the criticality weight and b_{ij} is the useful skew budget between sinks i and j. Skew budgets are assumed to be zero in this paper without loss of generality. The objective function can be rewritten as follows

$$\Phi\,(S) \quad = \quad \sum_{i>j} w_{ij}\,(s_i - s_j)^2 = \frac{1}{2} S^T Q S \tag{13}$$

where Q is the $n \times n$ Laplacian. The Laplacian is $Q = (q_{ij})$ where q_{ij} has a weight $-w_{ij}$ if $i \neq j$ and the diagonal entry q_{ii} is equal to the sum of the adjacent weights, $\sum_{j=1}^{n} w_{ij}$. For the remainder of the paper, all skew pairs are weighted equally, $w_{ij} = 1$. In order to minimize total skew, an iteration of SQP can be formulated with the following quadratic programming subproblem

$$\min \quad \Phi(S) = \sum_{i>j} (s_i - s_j)^2 \tag{14}$$

$$s.t. \quad D + G\Delta = S \tag{15}$$

$$P_{cur} + \beta\delta \leq P_{bnd}. \tag{16}$$

The new SQP formulation has several advantages over SLP. First, the objective function in the QP subproblem is positive semi-definite so the QP subproblem is convex and easily solvable. Assuming n clock sinks, the QP subproblem has $2n$ fewer constraints than the LP subproblem and two fewer variables due to the elimination of auxiliary variables, s_{max} and s_{min}. If constraint conditions are not enforced between all pairs, the QP subproblem can have up to n^2 fewer constraints than the LP subproblem. This reduction in problem complexity actually makes solving the quadratic program faster than solving the linear program in many cases. Last, but not least, the SQP method tends to converge more quickly than SLP methods.

5. ROBUSTNESS ANALYSIS

The quadratic objective function presented in the previous section is a heuristic to shape the skew wall for more robust clock trees. To accurately quantify the sensitivity of the clock trees to varying process parameters and to show the benefits of the quadratic objective function over traditional maximum skew, statistical analysis is performed using Monte Carlo simulations on the optimized trees.

Device variability can contribute significantly to the variation of clock trees due to the fluctuation of gate capacitance and drive strength. Both the gate capacitance and drive strength are varied proportionally to L_{eff} variation in these experiments. L_{eff} variation has a correlation distance of approximately 100-300μm beyond which it is independent [5]. Since clock sinks and buffers are typically distributed throughout a die, the variations of these devices are assumed to be independent.

This work also considers two physical parameters for metal variation: height (H) and width (W). CMP variation has a larger correlation distance than L_{eff} and is on the order of 3.5mm [5]. The tolerance for the range of interconnect density during the fill process is user-defined. Due to these considerations, the interconnect variation is assumed to be fully correlated in this paper. This does not exclude other more advanced models from being used. A simple extraction model is used for wire resistance and capacitance. Unit resistance is calculated using the cross sectional area and the square resistivity. Capacitance estimation uses a parallel plate model for inter-layer and sidewall capacitance and a term for fringe effects. The method can incorporate more accurate 3-D extraction models.

All variation in this work is assumed to be Gaussian, but Monte Carlo simulations can incorporate more advanced models when they are available. The nominal process values and variations are from a typical 90nm process [11]. The parameters used are: metal height, $280nm \pm 15\%$; metal width $175nm \pm 32nm$; and L_{eff}, $53nm \pm 16.7\%$. Other parameters such as dielectric thickness, spacing, resistivity and device threshold can also be considered, but the parameters used are several of the most significant sources of variation.

The expected value and standard deviation of the max-

imum skew in the Monte Carlo simulations quantifies the robustness of clock trees. The statistical measurements reflect the process induced skew that is evident throughout a large sampling of manufactured chips.

6. EXPERIMENTAL RESULTS

We implemented both the SLP and SQP methods in C++ on Linux. Buffers are restricted to $12 - 64\times$ minimum size with the ratio of the output to input stage fixed at four. Wires are restricted to $1 - 4\times$ minimum width. Continuous sizes are permitted since greedy discretization heuristics do not seem to affect solution quality significantly. The benchmarks were created using deferred merge embedding (DME) [4] and initially have zero deterministic skew with minimum size wires and no buffers. Buffers of the smallest allowed size were then inserted according to the method of [19] such that the path to every sink has an equal number of buffers and each buffer initially has a maximum load capacitance of $250fF$ for slew and reliability. The addition of the buffers leads to non-zero deterministic skew for the clock tree, but it is an adequate starting point for further optimization. The number of buffers inserted ranged from 27 to 1130 and the number of wire segments ranged from 245 to over 10,000. This means that the LP and QP subproblems had up to 11,000 variables. All LP and QP subproblems were solved using the COIN linear program solver, CLP [8]. It is not widely known, but CLP can solve quadratic minimization problems.

As shown in Section 3, the adaptive confidence method results in significant improvements over a static confidence value. Therefore, the comparison of SLP and SQP is done with both methods using adaptive confidence.

Results on the set of clock tree benchmarks are shown in Table 2 for a capacitance (power) bound of 50%. The deterministic skew of all benchmarks is slightly worse when using the quadratic objective. However, the SQP-optimized trees have a better expected skew (μ) and standard deviation (σ) than those created with SLP. Consequently, the 99.8% quantile, $\mu+3\sigma$, skew is an average of 35% better when using SQP. This is due to many sink pairs having near critical skew in the SLP case. The deterministic skew histograms for all worst pairs of sinks for SLP and SQP are shown in Figure 2. SLP has over 7,000 skew pairs within $1ps$ of the most critical whereas SQP has only 12 pairs within $1ps$ of the most critical. The net statistical effect is shown by the probability distributions of the maximum skews in the same figure. Both the mean and standard deviation of the SLP approach are much larger than with SQP.

Other capacitance bound values provide insight into how capacitance (power) is allocated to critical skew pairs. Figure 3 shows the capacitance-skew trade-off curves for benchmark r1. The deterministic skew of the SQP approach is consistently more than SLP, but the difference is least at the low and high capacitance bounds. The 99.8% quantile skew of SQP, however, is consistently improved as additional power is allocated. SLP, on the other hand, nears zero deterministic skew and does not improve the statistical skew beyond approximately $40ps$. At very tight power bounds, minimizing the deterministic maximum skew gives similar results to minimizing the total squared skew, because the skew wall has not been formed yet.

On average, the CPU time of the SQP approach is 34.4% less than the SLP approach. This is primarily due to SQP

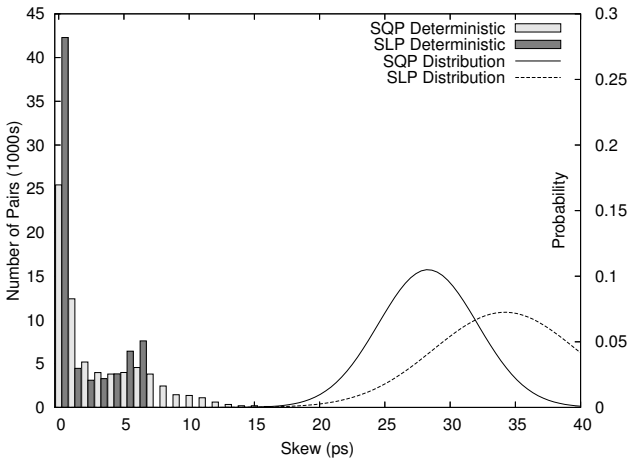

Figure 2: Results of SLP and SQP on r1.

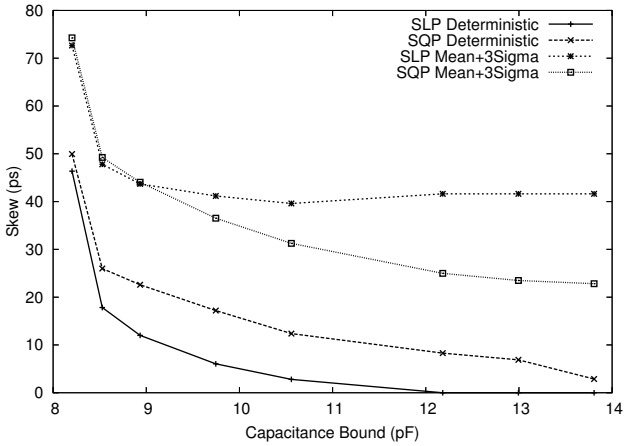

Figure 3: Benchmark r1 capacitance-skew trade-off.

requiring fewer iterations, on average, to converge on the final solution. In the 50% power bound case, SLP on the largest benchmark takes over 2.6 hours, whereas SQP takes only 1.38 hours. With more accurate timing models that require more run-time, the decrease in the number of iterations will become more apparent. Many optimizations were run with power bounds ranging from 10%–80% of minimum size for all benchmarks. The collective run-times were then averaged and fit against the number of buffers and wires to determine empirical computational complexities. The complexity of SLP is $O(n^{3.04})$ and SQP is $O(n^{2.59})$.

The previous results of the SLP and SQP approaches are not provably optimal, but the results are better than general purpose non-linear programming (NLP) optimization using IPOPT [8] given the limitations on run-time. IPOPT uses roughly $10,000\times$ the CPU time and is unable to match the quality of results. For the smallest benchmark, s1423, with a 10% capacitance bound, IPOPT was unable to converge on a solution within 100 iterations (6 hours) whereas SLP and SQP found slightly better solutions in 20 iterations (4 seconds) and 11 iterations (2 seconds), respectively.

Table 2: Optimization results for SLP and SQP with 50% power bound over minimum size solution.

			SLP					SQP				
Benchmark	Buffers	Wires	Det. Skew	μ (ps)	σ (ps)	$\mu+3\sigma$ (ps)	CPU (s)	Det. Skew	μ (ps)	σ (ps)	$\mu+3\sigma$ (ps)	CPU (s)
s1423	27	245	0.0	17.7	4.9	32.3	2.7	0.5	12.8	2.6	20.7	2.8
s5378	68	585	0.0	22.1	4.9	37.0	8.9	12.0	20.1	3.3	29.9	9.6
s15850	215	1917	0.0	32.3	6.5	51.9	152.4	13.3	30.3	5.0	45.2	80.4
r1	101	853	0.0	25.2	5.5	41.6	22.1	8.3	17.2	2.6	25.0	19.6
r2	223	1963	0.4	31.3	6.0	49.4	193.9	18.9	27.2	3.6	37.8	103.2
r3	324	2822	0.5	34.2	5.5	50.7	281.6	11.6	28.3	3.8	39.9	160.4
r4	707	6200	0.7	43.2	6.2	61.8	1993.5	11.7	31.7	4.0	43.5	1529.5
r5	1130	10061	1.9	46.2	5.8	63.7	9387.8	19.4	38.1	4.4	51.3	4981.7
p1	94	792	0.0	20.2	4.1	32.6	19.6	10.3	18.7	2.7	26.9	21.2
p2	211	1606	0.0	32.6	6.0	50.7	101.0	6.5	23.7	3.9	35.4	127.7
Average Improvement (%)									24.1	58.9	35.0	34.4

7. CONCLUSIONS & FUTURE WORK

This paper presented two major improvements to sequential programming for skew optimization using buffer and wire sizing. First, we showed that a simple adaptive confidence region metric can dynamically adjust the valid range of the sequential programming subproblem to improve solution quality while keeping the CPU time low. Second, we presented a new subproblem for sequential programming that uses total skew squared instead of the traditional maximum skew to obtain more robust results with less CPU time. On average, the new objective requires 34.4% less CPU time and produces clock trees with 24.1% lower expected skew and 58.9% lower standard deviation with the same power budget.

Our future work intends to investigate larger trees, examine the effects of useful skew solution quality, and directly optimize parametric yield.

8. REFERENCES

[1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical clock skew analysis considering intra-die process variations. In *ICCAD*, pages 914–920, 2003.

[2] C. Albrecht, A. B. Kahng, B. Liu, I. Mandoiu, and A. Zelikovsky. On the skew-bounded minimum buffer routing tree problem. In *SASIMI*, pages 250–256, 2001.

[3] X. Bai, C. Visweswariah, P. N. Strenski, and D. J. Hathaway. Uncertainty-aware circuit optimization. In *DAC*, pages 58–63, 2002.

[4] K. Boese and A. Kahng. Zero-skew clock routing trees with minimum wirelength. In *ASIC Conf.*, pages 1.1.1–1.1.5, 1992.

[5] Y. Cao, P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. Design sensitivities to variability: Extrapolations and assessments in nanometer VLSI. In *ASIC/SOC*, pages 411–415, September 2002.

[6] T.-H. Chao, Y.-C. Hsu, and J. Ho. Zero skew clock net routing. In *DAC*, pages 518–523, 1992.

[7] C.-P. Chen, Y.-P. Chen, and D. F. Wong. Optimal wire-sizing formula under the elmore delay model. In *DAC*, pages 487–490, 1996.

[8] Computational infrastructure for operations research. http://www.coin-or.org.

[9] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, 39(7):945–951, July 1990.

[10] M. Hashimoto, T. Yamamoto, and H. Onodera. Statistical analysis of clock skew variation in H-tree structure. In *ISQED*, 2005.

[11] International technology roadmap on semiconductors. http://public.itrs.net, 2004.

[12] R. Kay and L. T. Pileggi. EWA: Efficient wiring-sizing algorithm for signal nets and clock nets. *IEEE Trans. on CAD*, 17(1):40–49, January 1998.

[13] I. S. Kourtev and E. G. Friedman. A quadratic programming approach to clock skew scheduling for reduced sensitivity to process parameter variations. In *ASIC/SOC*, pages 210–215, 1999.

[14] I.-M. Liu, T.-L. Chou, A. Aziz, and D. F. Wong. Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion. In *ISPD*, pages 33–38, 2000.

[15] B. Lu, J. Hu, G. Ellis, and H. Su. Process variation aware clock tree routing. In *ISPD*, pages 174–181, 2003.

[16] J. L. Neves and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 623–629, 1996.

[17] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[18] S. Pullela, N. Menezes, and L. T. Pillage. Reliable non-zero skew clock trees using wire width optimization. In *DAC*, pages 165–170, 1993.

[19] G. E. Tellez and M. Sarrafzadeh. Minimal buffer insertion in clock trees with skew and slew rate constraints. *IEEE Trans. on CAD*, 16(4):333–342, 1997.

[20] J.-L. Tsai, T.-H. Chen, and C. C. Chen. Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing. *IEEE Trans. on CAD*, 23(4):565–573, April 2004.

[21] K. Wang and M. Marek-Sadowska. Buffer sizing for clock power minimization subject to general skew constraints. In *DAC*, pages 159–164, 2004.