

Clock Scheduling for Power Supply Noise Suppression using Genetic Algorithm with Selective Gene Therapy*

Wai-Ching Douglas Lam, Cheng-Kok Koh
ECE, Purdue University
W. Lafayette, IN 47907-1285
{dougla, chengkok}@ecn.purdue.edu

Chung-Wen Albert Tsao
Cadence Design Systems
San Jose, CA 95134
tsao@celestry.com

Abstract

Simultaneous switching events in the clock lines and almost simultaneous switching events of sequential and combinational logic elements cause large $L \cdot di/dt$ and IR voltage variations in the power and ground network of a circuit. This is known as power supply noise and it affects the performance and reliability of the entire circuit. In this paper, we propose a genetic algorithm based clock scheduling approach for minimizing the number of simultaneous switching events such that the power supply noise is suppressed. We ensure that in any generation of the genetic algorithm process, there will be feasible clock schedules present in the gene pool by the use of gene therapy. Experimental results on benchmark circuits show an average reduction of 26.2% in the peak current, an average reduction of 37.9% in the current swing, and an average reduction of 46.2% in voltage variations in the power lines.

1 Introduction

As circuits become more complex, the load on the clock and power lines increases. The situation is worse in synchronous systems with zero clock skew, as all sequential elements and their ensuing combinational logic are triggered almost simultaneously, each element drawing current from the power lines or sinking current to the ground lines, when the clock switches. The presence of resistive (R) and inductive (L) parasitics in power and ground (P/G) pins and lines leads to IR and $L \cdot di/dt$ voltage variations, respectively, in the P/G lines. This is commonly known as power supply noise and it may cause logic failure or affect the performance of the circuit.

To counter these problems in high speed designs, several physical design techniques have been proposed: Sizing up the P/G lines to accommodate the large current peaks and to minimize the IR and $L \cdot di/dt$ voltage variations in these lines [10]; and deploying decoupling capacitors in the P/G lines [1, 15]. In this paper, we strike at the root of the problem: simultaneous or nearly simultaneous switching of circuit elements due to zero clock skew. We propose a genetic algorithm (GA) that generates a feasible clock schedule such that the power supply noise is suppressed. The clock schedule can be realized

*This work was supported in part by NSF under contract number CCR-9984553, SRC under contract number 99-TJ-689, and a DAC scholarship

by clock routers [7, 9] that control the arrival time to the clock sinks.

Clock skew scheduling is an area of active research; [13, 14, 4] specifically targeted at reducing peak currents caused by switching activity. In [4], a fine-grained incremental clock skew scheduling approach was used. In [13], scheduling was done via integer linear programming. The flip-flops were clustered into bins that are skewed to switch at different times. However, such an approach suffers from the limitation that flip-flops within the same bin still switch simultaneously. [14] performed fine-grained clock scheduling using GA, i.e., assigning clock signal arrival times to individual flip-flops.

The approach that we take in this paper is similar in spirit as [14], i.e., we utilize GA to determine a feasible clock schedule. In addition, we consider the problem of generating infeasible schedules resulting from the crossover of two feasible schedules or the mutation of a feasible schedule during the intermediate steps of the GA. This may cause the gene pool to be filled with infeasible clock schedules in subsequent generations, which in turn may reduce the probability of obtaining a high quality feasible clock schedule in the final gene pool.

In this work, we ensure that feasible clock schedules are present in every generation. We achieve this by making infeasible skew schedules feasible through *gene therapy*. To the best of our knowledge, Ref. [14] did not address this problem.

We experimented with three strategies of applying gene therapy: performing gene therapy only on selected genes in every generation; performing gene therapy only at the very last generation and performing gene therapy for all genes in every generation. Experimental results on benchmark circuits show that best results were obtained by performing gene therapy on selected genes in every generation. Using this strategy, we achieve an average reduction of 26.2% in the peak current, an average reduction of 37.9% in the current swing, and an average reduction of 46.2% in voltage variations in the power lines.

2 Preliminaries and problem formulation

In this paper, edge triggered flip-flops are used, as in [13, 14] because it represents the worst case scenario for current peaks as all switchings are synchronized around the clock edges. Consider a simple synchronous circuit using positive edge-triggered flip-flops as sequential elements under a single-phase clocking scheme. Suppose ff_i and ff_j are two sequentially adjacent flip-flops, with ff_i feeding into a purely combinational logic block, which in turn feeds ff_j . Let the clock arrival times to ff_i and ff_j be t_i and t_j respectively. In order to

prevent hold time violation, we must have:

$$t_i - t_j \geq t_{hold,max} - t_{pFF,min} - t_{logic,min} + \delta_l. \quad (1)$$

Similarly, to prevent setup time violation:

$$t_i - t_j \leq T - t_{pFF,max} - t_{logic,max} - t_{setup,max} - \delta_u. \quad (2)$$

In the two preceding inequalities, $t_{logic,max}$ and $t_{logic,min}$ are the maximum and minimum propagation delays through the combinational logic block; $t_{pFF,max}$ and $t_{pFF,min}$ are the maximum and minimum propagation delays through the flip-flop; and T is the clock period. To correctly latch in data, the amount of time that the data has to remain stable before and after the clock triggers the flip-flop are t_{setup} and t_{hold} , respectively. Designers may impose additional constraints to increase the robustness of circuits to process variations. To be exact, we can add a safety margin $\delta_l (\geq 0)$ to the lower bound constraint in Eqn. (1) and subtracting a safety margin $\delta_u (\geq 0)$ to the upper bound constraint in Eqn. (2). This would force the clock schedules to be assigned further away from the extreme limits where hold or setup violation would occur.

In this work, the problem that we address can be stated as follows: *Given the skew constraints of a system, determine a non-zero clock skew schedule such that the power supply noise is minimized without violating any skew constraints.*

If a zero clock skew schedule is used, all flip-flops will trigger almost simultaneously. This will cause a large current to flow in the P/G line and will cause P/G noise due to resistive and inductive parasitics in P/G pins and lines. Therefore zero skew clock schedules are not desired in this context.

2.1 Constraint graph and feasibility

In this section, we briefly review the concept of clock scheduling [11] and also show the relationship between the feasibility of any given clock schedule and the constraint graph [8].

For simplicity, we use $l_{i,j} \leq skew_{i,j} = t_i - t_j \leq u_{i,j}$ to represent lower- and upper-bound skew constraints between ff_i and ff_j . We also denote an inequality $a \leq x \leq b$ as $x \in [a, b]$. $C = \{t_i - t_j \in [l_{i,j}, u_{i,j}]\}$ denotes the set of skew constraints for all sequentially adjacent flip-flops ff_i and ff_j .

To capture all the skew constraints for all pairs of flip-flops in a system, a constraint graph $G_C = (V, E)$ is used [12, 2, 3]. The flip-flop corresponds to a vertex in the constraint graph; the lower and upper bound constraint correspond to two directed edges $e_{i,j}$ and $e_{j,i}$, respectively. The weight of $e_{i,j}$, denoted by $w_{i,j}$, is $-l_{i,j}$, and the weight of $e_{j,i}$, denoted by $w_{j,i}$, is $u_{i,j}$. It is proved in [2] that feasible clock schedules exist if and only if G_C contains no negative cycles, which can be verified in polynomial time using the Bellman-Ford algorithm [2].

While the absence of negative cycles in G_C indicates the existence of a feasible clock schedule, G_C itself contains the set of skew constraints that determine the feasibility of any given clock schedule. Let the clock arrival times specified by a clock schedule be (t_1, \dots, t_N) , where N is the total number of flip-flops. We label the vertices of G_C with the corresponding clock arrival times in the given schedule. If G_C does not contain negative cycles, the given clock schedule is *feasible* if and only if the labels of the vertices satisfy the following condition:

$$t_i \leq t_j + w_{j,i} \quad \text{for all } e_{i,j} \in E. \quad (3)$$

As an example, consider a simple synchronous circuit with three positive edge-triggered flip-flops, ff_1 , ff_2 and ff_3 as sequential elements under a single-phase clocking scheme. The flip-flops have the following constraints:

1. $t_1 - t_2 \in [-10, 4]$,
2. $t_1 - t_3 \in [-6, -2]$, and
3. $t_2 - t_3 \in [2, 3]$.

These constraints form the G_C shown in Figure 2.1.

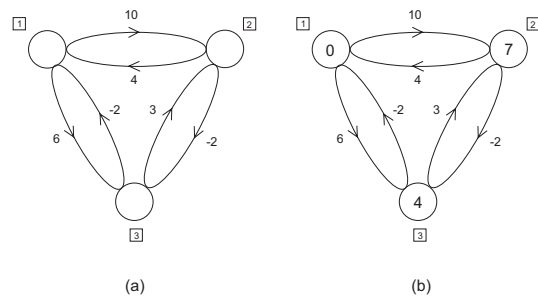


Figure 1: (a) Constraint graph. (b) Clock arrival times placed in vertices

Consider a clock schedule $(0, 7, 4)$. We label the vertices of G_C accordingly. It can be easily verified that the labels of the vertices satisfy all skew constraints in G_C , namely:

1. Between ff_1 and ff_2 :
 $7 \leq 0 + (10) \quad \text{and} \quad 0 \leq 7 + (4)$
2. Between ff_2 and ff_3 :
 $4 \leq 7 + (-2) \quad \text{and} \quad 7 \leq 4 + (3)$
3. Between ff_1 and ff_3 :
 $4 \leq 0 + (6) \quad \text{and} \quad 0 \leq 4 + (-2)$

This illustrates that the clock schedule $(0, 7, 4)$ is feasible with respect to G_C .

3 Genetic Algorithm

In this paper, we utilize a GA based approach to generate feasible clock schedules that minimizes the power supply noise. Power supply noise consists of two different components, namely IR and $L \cdot di/dt$ voltage drops. Reduction of IR voltage drops is done by minimizing the peak current. Reducing the $L \cdot di/dt$ voltage drops requires a minimization of the steepest slopes in the current profile. By setting these two components as fitness criteria in the evaluation function (required by any GA to perform gene selection), the objective of power supply noise minimization can be achieved. We shall present the evaluation function in Section 3.2.

However, the evolutionary and random nature of GAs will often produce genes that represent feasible clock schedules. Performing crossover or mutation on genes with feasible clock schedules may produce clock schedules that are infeasible. Retaining too many genes with infeasible clock schedules in the gene pool and using them in future evolutions may cause hegemony of such genes. When this occurs,

there is a possibility that there would be no feasible clock schedules available in the gene pool when the GA completes.

It is important to note that by simply forcing a low fitness factor on genes with infeasible clock schedules to discourage the GA to retain such genes does not solve the problem because it still does not guarantee the availability of feasible clock schedules in any generation. Therefore it is advantageous to perform gene therapy to control the population of genes that have infeasible clock schedules in a gene pool.

3.1 Gene therapy

Our main contribution in this paper is that if given a set of skew constraints that allows a feasible clock schedule to be obtained, we can perform gene therapy to doctor any gene with infeasible clock schedule into one that is feasible.

From the feasibility conditions given by Eqn. (3), it is obvious if a given schedule is infeasible, the following inequality must hold for some $e_{i,j} \in E$,

$$t_i > t_j + w_{j,i}. \quad (4)$$

When this occurs, the gene therapy process updates the label of vertex i such that

$$\tilde{t}_i = t_j + w_{j,i}, \quad (5)$$

where \tilde{t}_i is the updated label of vertex i .

Note that the above condition (Eqn. (4)) and update process (Eqn. (5)) of the therapy are exactly the *relaxation* steps performed in a shortest path algorithm, e.g., the Bellman-Ford algorithm. In the following, we show that the process of performing gene therapy is equivalent to a shortest path algorithm. We add a supernode, denoted by s , with outgoing edges to all the vertices in G_C (Figure 3.1). The weight of the outgoing edge $e_{s,i}$ is the clock arrival time t_i specified in the given clock schedule. We denote the new constraint graph as \hat{G}_C .

Fact 1: Suppose the constraint graph G_C constructed from skew constraints has no negative cycles. Then, \hat{G}_C does not have negative cycles.

Since the supernode in \hat{G}_C has only outgoing edges, it is impossible for any new cycles (positive or negative) to be formed with the addition supernode and its outgoing edges to G_C . Therefore, if G_C has no negative cycles, then \hat{G}_C has no negative cycles. In other words, the shortest paths from s to all vertices in \hat{G}_C are well-defined. We assume in the sequel that G_C has no negative cycles.

Fact 2: The process of correcting the labels of the vertices in G_C via gene therapy using the relaxation steps will converge to a unique solution that dependent on both the initial clock schedule and the skew constraints.

This can be proved by showing that performing gene therapy is similar to finding the single-source shortest paths from s to all vertices in \hat{G}_C . Since the shortest-paths are well-defined as shown in Fact 1, the order of correction is insignificant.

Let \hat{t}_i denote the shortest path from s to f_i in \hat{G}_C . At the end of the single-source shortest path algorithm, with s being the source node, \hat{t}_i satisfies the following:

$$\hat{t}_i = \min\{w_{s,i}, \min_{e_{i,j} \in E} \hat{t}_j + w_{j,i}\}. \quad (6)$$

Now, from Eqn. (5), it is clear that \tilde{t}_i , the new label of vertex i after gene therapy, satisfies the following:

$$\tilde{t}_i \leq \min_{e_{i,j} \in E} \tilde{t}_j + w_{j,i}. \quad (7)$$

Moreover, \tilde{t}_i satisfies the following constraint:

$$\tilde{t}_i \leq t_i = w_{s,i}. \quad (8)$$

Eqn. (5) also implies that \tilde{t}_i satisfies the following:

$$\tilde{t}_i = \min\{w_{s,i}, \min_{e_{i,j} \in E} \tilde{t}_j + w_{j,i}\}, \quad (9)$$

which is exactly Eqn. (6). In other words, applying gene therapy to G_C is equivalent to applying a shortest-path algorithm to \hat{G}_C . Hence, a unique solution is guaranteed regardless of the order of updates performed on G_C using gene therapy.

Performing gene therapy on a feasible clock schedule has no effects on the schedule because the labels of the vertices will not be updated (satisfies all constraints). On the other hand, if updates are made by the relaxation steps, the labels of the vertices will converge to a new feasible clock schedule that is purely determined by the given clock schedule and the given skew constraints.

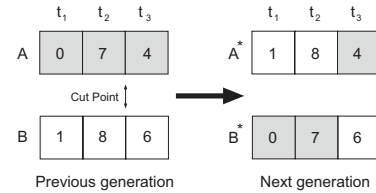


Figure 2: GA performing simple crossover.

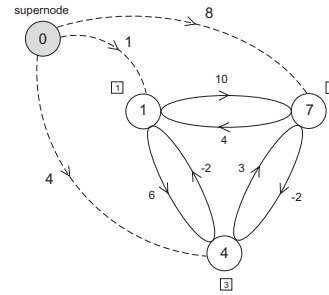


Figure 3: Addition of supernode and edges. The weight of dotted edges corresponds to the infeasible clock schedule (1, 8, 4). The values in the vertices represent the corrected arrival times of the new feasible clock schedule.

Consider the G_C in Figure 2.1 with a given set of skew constraints from Section 2.1. Let A and B denote two independent clock schedules from the gene pool. Gene A has clock arrival times (0, 7, 4) while gene B has clock arrival times (1, 8, 6). We can see that these two clock schedules are feasible.

Assuming that the GA selects these two genes to produce two new genes (A^* and B^*). When the crossover is performed as shown in Figure 3.1, the two new clock schedules (0, 7, 6) represented by gene A^*

and (1, 8, 4) represented by gene B^* are both infeasible. If we perform gene therapy on these two new clock schedules, the final clock arrival times will converge at (0, 7, 4) and (0, 7, 5) for gene A^* and gene B^* respectively. The new clock arrival times now satisfy Eqn. (3). Therefore, the two previously infeasible clock schedules become feasible.

3.2 Evaluation function and current waveforms

All genes in the gene pool must have an associated fitness value so that the GA solver can select the appropriate genes. In this context, a clock schedule that has a low power supply noise would have a higher fitness value. To characterize the power supply noise with any particular clock schedule, we utilize the clock arrival times to approximate the current profile and determine the amount of power supply noise based on that approximation.

When a flip-flop is triggered, we can observe the current waveform in a current-versus-time graph. From numerous HSPICE simulations of various flip-flop and combinational gate implementations, we observe that the current drawn by a flip-flop or a combinational logic gate can be represented by a piecewise linear waveform [14]. In this paper we use triangular waveforms to approximate current profiles.

To capture the currents due to flip-flops and combinational logic gates, we assume that the sequential circuit follows a staged structure, i.e., a set of flip-flops feeds into a network of combinational logic gates and the outputs are clocked in by another set of flip-flops. Consider a combinational logic gate k whose fan-in cone includes two input flip-flops, ff_i and ff_j . Suppose that ff_i is the reference flip-flop, and we use a *non-zero* clock skew schedule such that the clock arrival time of ff_j is later than that of ff_i and it does not violate the skew constraint between ff_i and ff_j . Let the shortest propagation delay from ff_i and logic gate k be t_{ik} and that from ff_j and logic gate k be t_{jk} . By using the shortest propagation delay, we capture all switching activities, including glitches.

To construct the current waveform, we first create two flip-flop waveforms for each flip-flop, each waveform separated by $T/2$. One waveform represents the current drawn by the flip-flops on the rising edge of the clock while the other represents the current drawn on the falling edge. Assuming that the combinational logic waveform translates rigidly with the clock arrival times to their input flip-flops, we create two identical combinational logic waveforms, each delayed by $t_i + t_{ik}$ and $t_j + t_{jk}$. We then take the waveform that envelopes the two individual combinational logic waveforms (Figure 4).

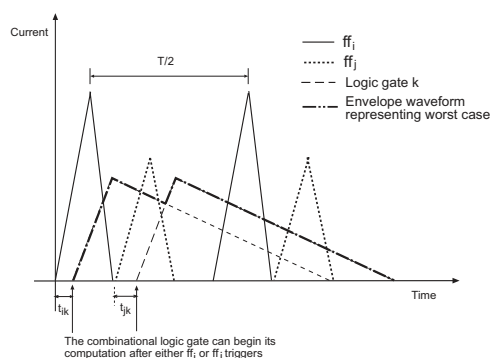


Figure 4: Current waveform of a combinational logic gate with two input flip-flops ff_i and ff_j .

For any particular clock schedule, the total current waveform due to the flip-flops and combinational logic gates is formed by summing up all the individual waveforms for flip-flops and envelope waveforms for combinational logic gates:

$$I_{total}(t) = \sum_{i=1}^K I_{comb_i}(t) + \sum_{i=1}^N I_{ff_i, rise}(t) + \sum_{i=1}^N I_{ff_i, fall}(t),$$

where I_{comb} , $I_{ff_i, rise}$, $I_{ff_i, fall}$ denote the current contributed by the combinational logic gate, the rising edge of flip-flops and the falling edge of flip-flops, respectively. N is the number of flip-flops and K is the number of combinational logic gates. Since all waveforms are represented by piecewise linear functions, the total current waveform can be computed by summing up all turning points, which can be done in linear time.

The peak current is simply the highest point in the current profile, denoted by I_{peak} while the max di/dt value is the steepest gradient in the current profile, denoted by di/dt_{max} . The fitness evaluation function is defined by:

$$F = \alpha(R \cdot I_{peak})^{-1} + \beta(L \cdot di/dt_{max})^{-1},$$

where α and β are weighing factors used to vary the sensitivity of the evaluation function, and R and L are parasitic values of the P/G package. Therefore, a clock schedule that has a low peak current and a low di/dt_{max} value would have a high fitness value.

3.3 Implementation issues

The algorithm begins by generating an initial population with genes that represent randomly generated clock schedules. We then perform gene therapy on all infeasible clock schedules in the initial population such that all genes now represent feasible clock schedules. The total current waveform (including combinational logic block) of every clock schedule in the population is then formed. A fitness value, based on the total current waveform, is assigned to every gene by the evaluation function. New genes are produced via crossover and mutation in the GA. For the crossover process, the crossover point is selected at random. Mutation is done by randomly varying the clock arrival times of one or more flip-flops chosen at random within the same clock period.

If there are infeasible clock schedules in subsequent generations, we perform gene therapy on some of these undesirable genes. Although it seems favorable to apply gene therapy on all infeasible clock schedules to make them all feasible, this would interfere with the GA's natural selection process. Therefore we arbitrary chose to apply gene therapy only on the genes that have infeasible clock schedules which are ranked the lowest 10% based on their fitness values. This will allow genes with infeasible clock schedules that have high fitness values to remain in the gene pool such that the resulting gene might have a high fitness value and also represent a feasible clock schedule.

The process of evolution and performing gene therapy is repeated until the number of user-specified generations is reached. With our approach, it is guaranteed that there will be genes that represent feasible clock schedules in every generation. This gives us the freedom to experiment with different types of crossover and mutation methods to obtain the most optimal solution.

For comparison purposes, we performed gene therapy via three different strategies: performing gene therapy only on selected genes in

every generation (lowest 10%); performing gene therapy on genes that have infeasible clock schedules only in the very last generation and performing gene therapy on all genes that have infeasible clock schedules in every generation. The results are shown in Section 4.

3.4 Complexity analysis

Here, we analyze the time complexity involved in correcting and evaluating one gene. Using a constraint graph $G_C = (V, E)$ to represent all skew constraints, we perform gene therapy to update the labels of the vertices. Therefore to update all vertices in one iteration, it takes $O(E)$. The update process will continue until there are no changes of the vertex labels. If it takes D iterations, our gene therapy process would therefore take $O(|E| \cdot D)$. When G_C has no negative cycles and happens to be sparse, D is a small constant [11, 5]. Therefore, the time complexity for gene therapy is $O(|V| = N)$, where N is the number of flip-flops.

Assume that there are at most p turning points used to store the worst case current waveform for a flip-flop or combinational logic gate. Constructing the total worst case current waveform, requires $O((N + K) \cdot p)$, where N and K are the number of flip-flops and combinational logic gates respectively.

4 Experiments and results

The proposed power supply noise reduction algorithm have been implemented in C++ and tested on six ISCAS89 benchmark circuits on a Sun UltraSparc-II. Given a circuit in Berkeley Logic Interchange Format, we first map the circuit to a TSMC $0.25\mu\text{m}$ cell library to obtain a HSPICE netlist. PathMill is then used to obtain the longest and shortest path delays for formulating the skew constraints (Eqns. (1) and (2)). The parameters used to approximate the waveform for the algorithm are extracted from HSPICE simulations of the cell library used. Then we apply the proposed algorithm to generate a clock skew assignment. With the optimized clock schedule, we perform PowerMill simulations on the benchmark circuits using the same 1000 randomly generated input vectors to determine the peak current, current and voltage swing reduction. The P/G line is assumed to have R and L parasitic values of 10Ω and 10nH , respectively, which are typical for the technology used [6].

The simulation results based on the three different gene therapy strategies used are tabulated in tables (1-3). In each table, optimization A refers to performing gene therapy on selected genes that have infeasible clock schedules; optimization B refers to performing gene therapy on genes that have infeasible clock schedules only in the very last generation and optimization C refers to performing gene therapy on all genes with infeasible clock schedules in every generation.

From the results, we see that there is a 6–48% reduction in the peak current, a 1–56% reduction in the current swing and a 8–73% reduction in voltage variations in power lines, using the selective gene therapy approach. Overall, the other two approaches give less reduction when compared to the selective gene therapy approach. We are keen on getting a fair comparison to [14], however, duplicating the results from their approach is difficult if not impossible because of several reasons. The technology level used was not mentioned in the paper. Secondly, a different simulator (PPP) was used to obtain the current profiles. Lastly, the parasitic inductance and resistance values used in their simulations are unknown.

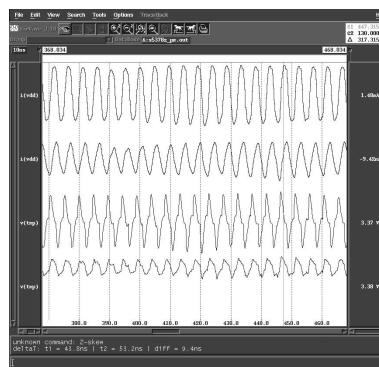


Figure 5: PowerMill simulation results of benchmark circuit s5378.

Figure 5 shows the current and voltage waveforms of the power network. In the figure, the first and the second waveforms are the current plots before (zero skew) and after optimization A, respectively. The third and fourth waveforms are the voltage plots before (zero skew) and after optimization, respectively.

5 Summary and Conclusions

A methodology has been presented to minimize the power supply noise via clock scheduling using GA. The main emphasis of the approach is the idea of using gene therapy to convert genes with infeasible clock schedules to genes with feasible clock schedules, such that valid solutions can be ensured to be available in the very last generation of the GA.

It is important to note that the concept of gene therapy can also be utilized in other algorithms, such as simulated annealing, where the availability of a feasible clock schedule is not guaranteed at the final or intermediate steps of the algorithm.

References

- [1] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.
- [2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 25.5, pages 539–543. The MIT Press, 1990.
- [3] R. B. Deokar and S. S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 407–410, 1994.
- [4] W.-C. D. Lam, C.-K. Koh, and C.-W. A. Tsao. Power supply noise suppression via clock skew scheduling. In *Proc. of the ISQED*, pages 355–360, 2002.
- [5] Y-Z Liao and C.K. Wong. An algorithm to compact a vlsi symbolic layout with mixed constraints. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 62–69, 1983.
- [6] A. Odabasioglu, M. Celik, and L. T. Pileggi. Prima: passive reduced-order interconnect macromodeling algorithm. In *Proc. Int. Conf. on Computer Aided Design*, pages 58–65, 1997.
- [7] M. Seki, K. Inoue, K. Kato, K. Tsurusaki, S. Fukasawa, H. Sasaki, and M. Aizawa. A specified delay accomplishing clock router using multiple layers. In *Proc. Int. Conf. on Computer Aided Design*, pages 289–292, 1994.

Table 1: Peak current reduction.

Circuit	Number of clock pins	Peak Current (mA)						
		Before optimization	After optimization A	% reduction	After optimization B	% reduction	After optimization C	% reduction
s208	8	1.61	1.52	5.6	1.60	0.6	1.60	0.6
s420	16	2.87	2.27	20.9	2.49	13.2	2.49	13.2
s1423	74	14.2	9.44	33.5	10.20	28.2	10.09	28.9
s5378	163	28.60	21.23	25.8	21.91	23.4	21.34	25.4
s9234	135	17.03	13.06	23.3	13.61	20.1	12.80	24.8
s15850	597	58.07	30.18	48.0	47.47	18.3	48.04	17.3
Average				26.2		11.1		18.4

Table 2: Current swing reduction.

Circuit	Current swing (mA)						
	Before optimization	After optimization A	% reduction	After optimization B	% reduction	After optimization C	% reduction
s208	1.55	1.54	0.6	1.55	0.01	1.55	0.01
s420	3.27	2.12	35.2	2.52	22.9	2.51	23.2
s1423	16.40	8.38	48.9	10.32	37.1	10.39	36.6
s5378	31.02	16.29	47.5	17.68	43.0	16.49	46.8
s9234	19.03	11.49	39.6	12.63	33.6	11.87	37.6
s15850	59.65	26.57	55.5	41.82	29.9	42.63	28.5
Average			37.9		27.8		28.8

Table 3: Voltage swing reduction.

Circuit	Voltage swing (V)						
	Before optimization	After optimization A	% reduction	After optimization B	% reduction	After optimization C	% reduction
s208	0.12	0.11	8.3	0.11	8.3	0.12	0.01
s420	0.13	0.10	23.1	0.11	15.4	0.11	15.4
s1423	0.42	0.14	66.7	0.20	52.4	0.16	61.9
s5378	0.60	0.28	53.3	0.29	51.7	0.30	50.0
s9234	0.36	0.17	52.8	0.18	50.0	0.18	50.0
s15850	0.66	0.18	72.7	0.29	56.1	0.30	54.5
Average			46.2		39.0		38.6

- [8] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Graph algorithms for clock schedule optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 132–136, 1992.
- [9] H. Shin and A. Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: MIGHTY. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(6):942–955, 1987.
- [10] H. H. Su, K. Gala, and S. Sapatnekar. Fast analysis and optimization of power/ground network. In *Proc. Int. Conf. on Computer Aided Design*, pages 477–480, 2000.
- [11] T.G. Szymanski. Computing optimal clock schedules. In *Proc. Design Automation Conf.*, pages 399–404, 1992.
- [12] C.-W. A. Tsao and C.-K. Koh. UST/DME: A clock tree router for general skew constraints. In *Proc. Int. Conf. on Computer Aided Design*, pages 400–405, 2000.
- [13] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska. Clock skew optimization for ground bounce control. In *Proc. Int'l Conf. on Computer Aided Design*, pages 395–399, Nov. 1996.
- [14] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli. Clock-skew optimization for peak current reduction. In *Proc. Int'l Symp. on Low Power Electronics and Design*, pages 265–270, Aug. 1996.
- [15] Shiyou Zhao, Kaushik Roy, and Cheng-Kok Koh. Decoupling capacitance allocation for power supply noise suppression. In *Proc. 2001 International Symposium on Physical Design*, pages 66–71, 2001.