

# Power Supply Noise Suppression via Clock Skew Scheduling \*

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

Chung-Wen Albert Tsao  
Celestry Design Technologies,  
San Jose, CA 95112  
tsao@celestry.com

## Abstract

*Simultaneous switching events in the clock lines and the signals passing through sequential and combinational logic elements cause large  $L \cdot di/dt$  and  $IR$  voltage variations in the power and ground network. This is known as power supply noise and it affects the performance and reliability of the entire circuit. In this paper, we propose an algorithm that performs clock skew scheduling to minimize the number of simultaneous switching events such that the power supply noise is suppressed. Our approach establishes a direct relationship between current (drawn by a circuit element, sequential or combinational) and skew by the concept of envelope waveforms, using a graphical representation. We provide a graph-based scheduling approach to reduce the peak current and to minimize the difference between the current peaks and valleys such that the current profile of the entire circuit is smoothed. Our approach also guarantees that the resulting clock schedule does not violate setup and hold time constraints. Experimental results on benchmark circuits show an average reduction of 19.6% in the peak current, an average reduction of 38.7% in the current swing, and an average reduction of 47.4% 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/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 may cause logic failure or erratic behavior 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]; increasing the number of P/G pins [4]; and deploying decoupling capacitors in the P/G lines [1; 16]. 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 an algorithm that performs clock skew scheduling to minimize the number of simultaneous switching such that the power supply noise is suppressed. By reducing the current peak, it directly translates to

reduced  $IR$  voltage drops. Also, by reducing the difference between current peak and valleys, we smoothen the current-versus-time profile, and that indirectly leads to a reduction in  $L \cdot di/dt$  voltage variations.

Clock skew scheduling is an area of active research [5; 8; 11; 9; 3; 14; 13; 7; 15; 12]. It is worthy of note that [13; 14] specifically targeted at reducing peak currents caused by switching activity. In [13; 14], flip-flops are 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. The decision that led to the clustering of flip-flops stemmed from the assumption that fine-grained control of clock skew through routing was difficult [13; 14]. However, recent advances in clock routing [15; 12] indicated otherwise. To be fair, Ref. [14] also performed fine-grained clock scheduling using genetic algorithm. In an effort to reduce the run-time complexity of the genetic algorithm, however, Ref. [14] made an oversimplified assumption that clock scheduling would not affect the current waveform of the combinational logic gates.

In this work, besides performing fine-grain clock scheduling, i.e., assigning clock signal arrival time to individual flip-flops, we also consider the effect of clock scheduling on the current waveforms of the combinational logic gates. As in [3; 12], we use a graph-based approach to capture the skew constraints among the flip-flops. We also establish a direct relationship between current (drawn by a circuit element, sequential or combinational) and skew using a graphical representation, called *envelope waveform*. We reduce the peak and the difference between the peaks and valleys of the current waveform, such that the current profile of the entire circuit is smoothed.

## 2. Preliminaries

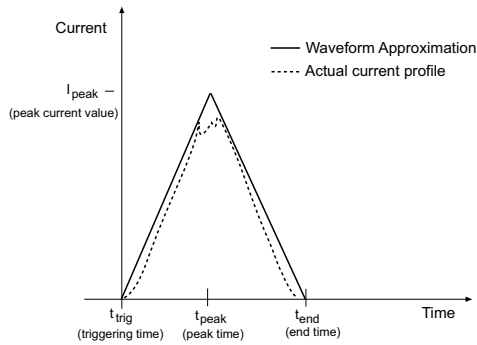
In this paper, edge triggered flip-flops are used, as in [13; 14] because it represents the worst case scenario for current peaks as all switchings are synchronized at the clock edges. Consider a simple synchronous circuit using positive edge-triggered flip-flops as sequential circuits under a single-phase 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:

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

Similarly, to prevent setup time violation:

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

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



**Figure 1: An example showing the approximation of the actual current profile.**

In the two preceding inequalities,  $t_{logic,max}$  and  $t_{logic,min}$  are the maximum and minimum 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.

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  $x \in [a, b]$ .  $\mathcal{C} = \{t_i - t_j \in [l_{i,j}, u_{i,j}]\}$  denotes the set of skew constraints for all sequentially adjacent flip-flops  $ff_i$  and  $ff_j$ .

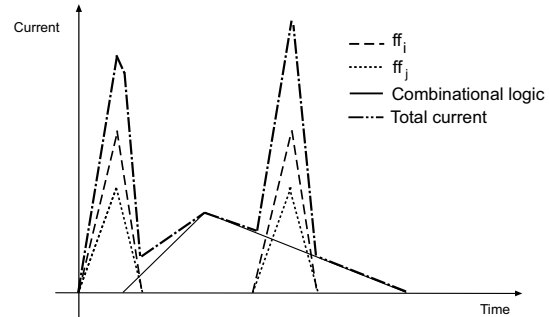
## 2.1 Current waveforms

When a flip-flop is triggered, we can observe the current waveform that displays the actual current drawn in a current-versus-time graph (Figure 1). From numerous HSPICE simulations of various flip-flop implementations, we observe that the current drawn by a flip-flop or a combinational logic gate can be represented by a triangular waveform [14]. The triangular waveform can be characterized by four parameters shown in Figure 1. They essentially define the turning points of the approximated waveform and are obtained from HSPICE simulations of various flip-flops and logic gates. If the need arises, piece-wise linear waveforms can be used to achieve better approximation by simply increasing the number of turning points. In this paper we use triangular waveforms to approximate current profiles.

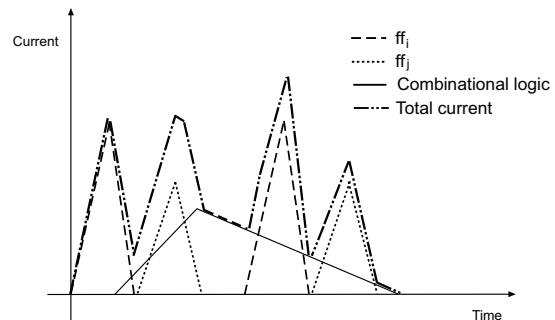
Consider a simple circuit that contains a single flip-flop ( $ff_i$ ) that feeds into a combinational logic gate and the output of the gate feeds into a flip-flop ( $ff_j$ ). Figure 2 shows the flip-flop current waveforms ( $ff_i$  and  $ff_j$ ) with the combinational logic waveform. Here we focus on the flip-flop current waveforms for this section and assume that the combinational logic waveform starts closely after  $ff_i$  triggers (details about the combinational logic waveform is presented in Section 2.2). Each flip-flop produces two waveforms, 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. The total current waveform is obtained by summing up all the individual waveforms.

With a *zero* clock skew schedule, we see that the peaks of the two flip-flop current waveforms overlap each other. As a result, two high current peaks near the rising and falling clock edges are observed from the total current waveform. Using the same circuit

example, now assume that we have a *non-zero* clock skew such that the clock arrival time of flip-flop  $ff_j$  is later than that of  $ff_i$  and it does not violate the skew constraint between  $ff_i$  and  $ff_j$ . Assuming that  $ff_i$  is the reference flip-flop (as for the rest of the paper), this translates to a right shift of the current waveform of  $ff_j$  by the skew amount (Figure 3). Since  $ff_i$  is the reference flip-flop, the waveform of  $ff_i$  and the corresponding combinational logic gate are not shifted. We see that the flip-flop current waveforms do not overlap as much when compared to the zero skew case. From these two examples, we demonstrate the relationship between skew and current by using a graphical representation and show how skew can be intentionally introduced such that the peak current can be minimized.



**Figure 2: An example showing the waveforms in one clock cycle with zero skew to the flip-flops.**

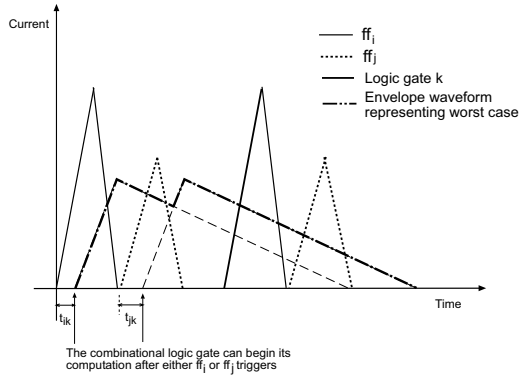


**Figure 3: An example showing the waveforms in one clock cycle with a non-zero skew to the flip-flops.**

## 2.2 Dealing with combinational logic

To capture the currents due to 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  $ff_j$ . Let the shortest path 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 path delay, we capture all switching activities, including glitches. To construct the current waveform, we first create two flip-flop waveforms like in the previous section, then 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).

From the example, we see that both the flip-flop current waveform and the combinational logic waveform would be skewed by



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

the delayed clock arrival time. We can extend this method of waveform construction to any combinational logic gate accordingly.

By forming the waveform in such a manner, we can capture all possible cases of switching events along the path of a stage, including current glitches that occur in many combinational logic gates.

### 2.3 Problem formulation

The total current waveform due to the flip-flops and combinational logic gates is formed by summing up all the individual envelope waveforms:

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

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. A piece-wise linear approach is used to compute the total current by summing up all turning points, which can be done in linear time.

The current causes P/G noise due to resistive and inductive parasitics in P/G pins and lines. In this work, the problem that we address can be stated as follows: *Given the skew constraints of a system, determine the clock schedule such that the power supply noise is minimized without violating any skew constraints.*

We minimize the power supply noise by targeting the two different component of noise, namely  $IR$  and  $L \cdot di/dt$  voltage drops. Reduction of  $IR$  voltage drops is done by reducing the peak current. By reducing the difference between the current peaks and valleys, we smoothen the current profile and this would indirectly translate to a reduction in  $L \cdot di/dt$  voltage drops.

## 3. Feasible skew range

It may seem that once all skew constraints (Eqns. (1) and (2)), for all flip-flops have been setup, we are allowed to select any skew values within those ranges. To counter this point, consider the example with the following local skew constraints between three adjacent flip-flops ( $ff_1$  feeds into  $ff_2$  while  $ff_2$  feeds into  $ff_3$ ):

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

Clearly, we see that zero skew is within these three local skew constraints. However, if we select  $skew_{,2} = 0 \in R_{1,2}$ , the other two skew constraints,  $t_1 - t_3 \in R_{1,3}$  and  $t_2 - t_3 \in R_{2,3}$ , cannot be satisfied simultaneously, and thus no feasible skew schedule can be found. Therefore, to obtain a valid skew schedule, we need to present the idea of *feasible skew range (FSR)*, so that we can schedule the flip-flop to be triggered at a specific time within the  $FSR$ . This has been explored in detail in [12]. It is summarized here to show how it can be applied to this context.

The *feasible skew range* between a pair of flip-flops defines the maximal skew values that can be assigned to the pair and still allow a feasible schedule for the entire circuit. To capture all the skew constraints for all pairs of flip-flops in a system, it is best to use a constraint graph  $G_C = (V, E)$  as in [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  $e_{i,j}$  is  $-l_{i,j}$ , and the weight of  $e_{j,i}$  is  $u_{i,j}$ . The upper bound of the implied skew constraint between  $ff_i$  and  $ff_j$  is equivalent to the shortest distance from  $ff_i$  to  $ff_j$ . Likewise, the lower bound of the transitively induced constraints between  $ff_i$  and  $ff_j$  is the negative of the shortest distance from  $ff_i$  to  $ff_j$ . When  $G_C$  contains no negative cycles, the shortest distance from  $ff_i$  to  $ff_j$ , denoted by  $d_{i,j}$ , is well-defined. It is proved in [2] that feasible skew schedules exist if and only if  $G_C$  contains no negative cycles.

Therefore, with the constraint graph  $G_C$  formed, we can compute a all-pairs-shortest-distance matrix  $\mathcal{D} = \{d_{i,j} : ff_i, ff_j \in \mathcal{FF}\}$  from  $G_C$  using the Floyd-Warshall algorithm to represent all feasible skew ranges. The maximum feasible skew range between  $ff_i$  and  $ff_j$ , under all the implicit and explicit constraints, can be simply expressed as  $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ . Hence, we also refer to  $\mathcal{D}$  as the  $FSR$  matrix.

### 3.1 Worst case current waveform

In this section, we show how  $FSR$ s are represented in current waveforms. Since skew is a relative measure between flip-flops, one flip-flop ( $ff_i$ ) will be treated as a reference and have  $FSR_{i,i} = [0,0]$ . All the other  $FSR$ s will be treated with respect to the reference flip-flop. This means that when we create the current waveforms, we only use the  $FSR$ s that are related to the reference flip-flop. In the  $FSR$  matrix, we will only be interested in all  $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ , where  $j \neq i$  and  $ff_j$  can be any other flip-flop. These  $FSR$ s will be in the row and column that correspond to the reference flip-flop in the  $FSR$  matrix.

To incorporate the  $FSR$  of a flip-flop in a waveform, we take the envelope waveform that bounds all the possible time locations where the individual waveforms are allowed to lie (Figure 5). The new envelope waveform can be represented by five parameters shown in the figure. By capturing the envelope waveform as a piecewise linear function, it facilitates the minimization process performed on the waveform (Section 4).

The envelope waveform for the combinational logic gate can be constructed in the same fashion since we assumed that the combinational logic waveform closely follows the triggering of the flip-flops. With  $FSR_{i,j}$ , we would construct the waveform as shown in Figure 6. We can see that the combinational logic waveform has " $FSR$ s" corresponding to the  $FSR$ s of its fan-in flip-flops and it is also stored as a piecewise linear function.

The total worst case current waveform is obtained by representing all the  $FSR$ s by envelope current waveforms and summing them together. We then perform our optimization procedure

to minimize power supply noise and to generate a feasible clock skew schedule. When the final clock schedule is obtained, all the  $FSR$ s would be reduced to specific values, and all the final individual envelope waveforms would transform back to waveforms similar to the ones we saw in Figure 4.

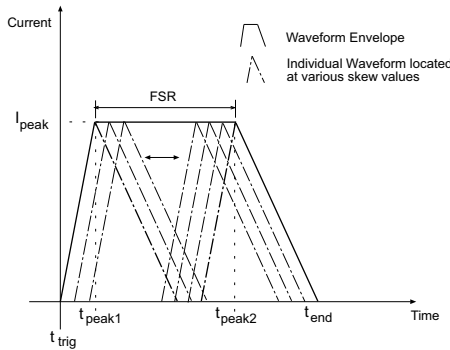


Figure 5: An example of an envelope waveform bounding all possible locations of individual waveforms.

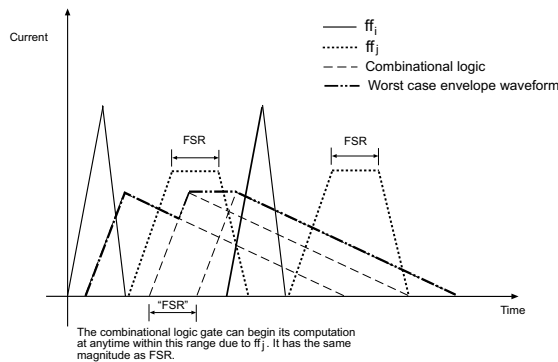


Figure 6: The same example as Figure 4, but now with the presence of a  $FSR$ .  $ff_i$  is the reference flip-flop.

## 4. Power supply noise suppression

In this section, we present the power supply noise suppression algorithm. Here, we assume a feasible clock schedule exists, i.e., no negative cycles exists in the constraint graph  $G_C$ . The summary of our approach is shown in Figure 4.3.

### 4.1 Peak current minimization

After obtaining the total worst case current waveform and the peak current, we perform a vertical slice at the peak time location. We then examine each individual envelope waveform (flip-flops and combinational logics) that intersects with the slice.

Suppose that the slice cuts through the envelope waveform of  $ff_j$ . This means that  $ff_j$  contributes to the peak current. Since the waveform is well defined by the turning points, we can easily obtain the gradient, current value and skew information of the contributing flip-flop at the point where the slice intersects.

Since  $I_{peak}$  of  $ff_j$  cannot be varied (value obtained by HSPICE simulations), the only way to reduce the contribution to the peak current is to shift the slopes of the waveforms left or right by reducing or tightening  $FSR$   $i,j$ . If the slice happens to cut through the right slope (positive gradient), we shift that slope left to reduce

its contribution to the total peak current (Figure 7). Shifting the slope left translates to a reduction in  $d_{j,i}$ , where  $ff_i$  is the reference flip-flop. Similarly, we shift the slope right to reduce the contribution if the slice cuts through the left slope (negative gradient), which leads to an increase in  $-d_{i,j}$ . If the cut intersects a flat portion of the waveform, the slope should be shifted in the direction that provides the most reduction using the least amount of skew change, and  $-d_{j,i}$  or  $d_{j,i}$  is changed accordingly.

For all cases, the greatest amount of shift allowed is the entire  $FSR$  of  $ff_j$ , i.e.  $(d_{i,j} + d_{j,i})$ . In other words, if the cut intersects an envelope waveform that has previously shifted the entire  $FSR$ , i.e.,  $FSR_{i,j} = [x, x]$ , no shift can be made for that waveform.

Since the “ $FSR$ s” of the envelope waveform of a combinational logic gate is related to the  $FSR$ s of the corresponding input flip-flops, whenever we tighten the  $FSR$  of a flip-flop, the corresponding “ $FSR$ ” of the combinational logic waveform must be tightened by the same amount as well. Similarly, if the slice happens to cut through an envelope waveform of a combinational logic gate, we perform the same procedure to shift the slopes and tighten the “ $FSR$ ” and the  $FSR$  of the corresponding flip-flop.

Of all envelope waveforms contributing to the current peak, the one that requires the least shift and produces the most current reduction is selected for  $FSR$  tightening. The motivation behind shifting a small amount each time is to prevent the  $FSR$  of any pair of flip-flops from being over-tightened that may, in turn, cause significant reductions of  $FSR$ s of other flip-flops. This will provide more flexibility when performing clock scheduling in subsequent iterations.

In Section 3, we have shown how the  $FSR$  of two flip-flops is defined by skew constraints between all flip-flops. Therefore, if any  $FSR$  is changed, we perform a quick update to the constraint graph  $G_C$ . The update is done by the use of *Incremental Skew Scheduling* technique in [12] and it is reviewed in Section 4.3.

### 4.2 Reduction of peak-and-valley difference

This optimization procedure is similar to the peak current minimization technique used in the previous section. It is used to prevent the lowering of the lowest valley of the total worst case waveform and is used when we have lowered the peak current to a minimum but not every  $FSR$  is tightened to a single value. With the total worst case waveform, we locate the lowest valley and perform a vertical slice. The envelope waveforms that intersect with the slice contribute to the lowest valley. Now instead of shifting the slopes that the slice intersects, we shift the opposite slopes and tighten the corresponding  $FSR$ s accordingly. And among all opposite slopes that we can shift, we select and shift the one that produces the least current reduction per unit shift. This will prevent the lowest valley from getting lower while continuing to tighten the remaining  $FSR$ s. The  $FSR$ s are updated using the incremental skew scheduling technique.

### 4.3 Incremental skew scheduling

It is shown in [12] that if  $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$  is changed to a subrange  $[x, y]$  such that  $-d_{i,j} \leq x \leq y \leq d_{j,i}$ , then the  $FSR$  matrix  $D$  can be updated (with no negative cycles created) as follows:

$$d_{k,l} = \min\{d_{k,l}, d_{k,i} - x + d_{j,l}, d_{k,j} + y + d_{i,l}\}, \forall 1 \leq k \neq l \leq n$$

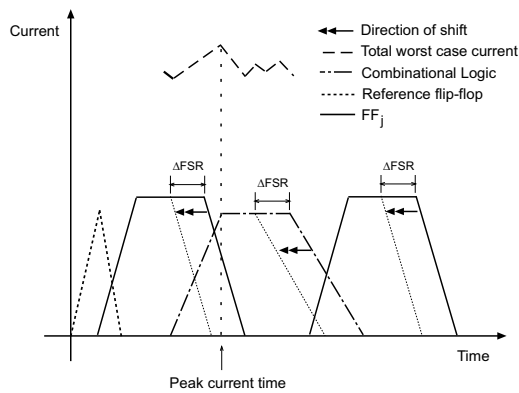
The use of incremental update of  $FSR$ s in this context is well suited because our approach begins with  $FSR$ s that allow a feasible skew schedule to be obtained, i.e., no negative cycles exists

in  $G_C$ . Also, whenever slopes are shifted in every iteration, we always tighten the corresponding  $F\mathcal{R}$  s to a subrange.

After updating  $G_C$ , the individual envelope waveforms for the flip-flops and combinational logics are then regenerated with the updated  $F\mathcal{R}$  s and a new current peak or valley can be found.

This process of determining the peak or valley, performing a slice, tightening the  $F\mathcal{R}$  , updating the skew and regenerating the waveforms is repeated until the new slice does not cut through any waveform that can be shifted, i.e., a waveform that has  $F\mathcal{R}_{i,j} = [x, y]$ , where  $x < y$ .

The remaining envelope waveforms that has  $F\mathcal{R}$  s that are not tightened to single values are those of flip-flops or combinational logic gates that do not contribute to the peak current nor the valleys. This means we are unable to alter the peaks or valleys further no matter how we change the remaining  $F\mathcal{R}$ s. Therefore, we make use of the remaining  $F\mathcal{R}$ s to distribute the skew evenly so that overlapping triggering events are further avoided.



**Figure 7: An example showing the case where the slice cuts through the right slope of  $ff_j$ . The slope of  $ff_j$  and the corresponding combinational logic waveform is shifted left by  $\Delta FSR$  .**

## 5. Complexity analyses

Let  $N$  and  $K$  denote the number of flip-flops and combinational logic gates respectively. For one iteration, the most expensive operation in our algorithm is the update of the  $F\mathcal{R}$  matrix  $\mathcal{D}$  after we tighten the  $F\mathcal{R}$  between flip-flops and it requires  $O(N^2)$ . From our experience, the algorithm will take about  $O(N)$  iterations to tighten every  $F\mathcal{R}$  to a single value. Therefore, the overall complexity is  $O(N^3)$ . However, if the constraint graph  $G_C$  is sparse such that the number of edges,  $m = |E|$  are proportional to the number of flip-flops  $N$ , i.e.,  $m = O(N)$ , we can reduce the complexity of the incremental scheduler as follows. Let  $P = \{F\mathcal{R}_{i,j,k} | k = 1, 2, \dots\}$  denote the  $F\mathcal{R}$  targeted for tightening in the  $k^{th}$  iteration. We can use a shortest path algorithm such as the Bellman-Ford algorithm [2] to obtain the new  $F\mathcal{R}$  s. When  $G_C$  is sparse, the complexity of Bellman-Ford is in practice  $O(c \cdot m) = O(N)$ , where  $c$  is a small constant [11; 6]. After making a change in the  $F\mathcal{R}$  , we update  $G_C$  with new edges or new edge weights between nodes  $ff_i$  and  $ff_{j,k}$  to reflect the  $F\mathcal{R}$  change. Hence, the number of edges will either increase by 2 or remain unchanged, depending on whether there are existing edges between  $ff_i$  and  $ff_{j,k}$ . Therefore, the number of edges is still  $O(N)$  after each  $F\mathcal{R}$  change.

1. Construct constraint graph  $G_C = (V, E)$  from skew constraints.
2. if  $G_C$  has negative cycles, return no solution and exit.
3. Build single source pairs-shortest distance matrix from  $G_C$ , using one flip-flop as reference.
4. Build the worst case current waveforms according to the  $F\mathcal{R}$ s.
5. **Perform Peak Current Minimization:**
  - a) Sum up the total worst case waveforms.
  - b) Determine magnitude and time location of the peak current.
  - c) Identify the waveforms that contribute to that peak.
  - d) Among all the waveforms that contribute to that peak, select one that provides a reduction with the least amount of skew changes and tighten the  $F\mathcal{R}$  accordingly such that the contribution is reduced or removed.
  - e) Perform incremental update for changes made in the  $F\mathcal{R}$ .
  - f) Rebuild total worst case waveform with updated  $F\mathcal{R}$ s.
  - g) **Repeat** steps 5(a) through 5(f) until,  $F\mathcal{R}$ s of flip-flops or combinational logic gates that contribute to the peak current are reduced to a single value.
6. **Perform Peak and Valley Difference Reduction:**
  - a) Determine magnitude and time location of the lowest valley.
  - b) Identify the waveforms that contribute to that valley.
  - c) Among all the waveforms that contribute to that valley, select one that provides a small reduction with the least amount of skew changes and tighten the  $F\mathcal{R}$  accordingly.
  - d) Perform incremental update for changes made in the  $F\mathcal{R}$ .
  - e) Rebuild total worst case waveform with updated  $F\mathcal{R}$ s.
  - f) Sum up the total worst case waveforms.
  - g) **Repeat** steps 6(a) through 6(f) until, All valleys have been targeted.
7. Make use of the remaining non-single value  $F\mathcal{R}$ s of flip-flops or combinational logic gates to distribute the skew evenly.

**Figure 8: Summary of our approach.**

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. Reconstructing the total worst case current waveform, after updating  $G_C$ , requires  $O((N + K)p)$  per iteration. Therefore, the overall complexity required to perform the incremental update due to  $O(N)$   $F\mathcal{R}$  changes and to reconstruct the worst case waveforms is  $O(N) \cdot O(N + (N + K)p) = O(N^2 + N(N + K)p)$ .

## 6. Experiments and summary

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 $\mu$ 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 (Table 1). 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 (Tables 2–4). The P/G package is assumed to have  $R$  and  $L$  parasitic values of  $10\Omega$  and  $10nH$ , respectively, which are typical for the TSMC 0.25 $\mu$ m technology used.

From the results, we see that there is a 5–36% reduction in the peak current, a 1–73% reduction in the current swing and a 5–89% reduction in voltage variations in power lines, using our approach.

The first optimization method of repeatedly targeting the peak allows the maximum reduction of the peak while the other areas are slightly reduced. This would lead to a reduction in  $IR$  voltage drops. Moreover, since the worst case waveform captures any possible glitches that may occur, our optimization method would also

**Table 1: Parameters used for waveform approximation.**

Type of element	$t_{trig}$ (ns)	$t_{peak}$ (ns)	$t_{end}$ (ns)	$I_{peak}$ ( $\mu A$ )
Flip-flop	0.0	0.5	1.5	300
NAND2	0.0	0.4	0.9	52
NAND3	0.0	0.4	0.8	65
NAND4	0.0	0.4	0.7	87
NOR2	0.0	0.5	1.2	89
NOR3	0.0	0.6	1.6	123
NOR4	0.0	0.5	1.7	180
Inverter	0.0	0.4	0.8	39

reduce those peaks that are caused by glitches (an unintended benefit of our optimization method). The second optimization method of performing clock scheduling to avoid the reduction of the valleys prevents the difference between the peaks and the valleys from increasing. This smoothens out the current profile and indirectly translates to a reduction in  $L \cdot di/dt$  voltage drops. Using these two optimization methods together, the power supply noise can be suppressed significantly as shown in the PowerMill simulation results in Figure 9. In the figure, the first and the second waveforms are the current plots for the power lines before (zero skew) and after optimization, respectively. The third and fourth waveforms are the voltage plots for the power lines before (zero skew) and after optimization, respectively.

**Table 2: Peak current reduction.**

Circuit	Number of clock pins	Peak Current (mA)		
		Before optimization	After optimization	% reduction
s208	8	1.61	1.52	5.6
s420	16	3.50	2.88	17.7
s1423	74	14.2	9.13	35.7
s5378	163	29.1	24.3	16.5
s9234	135	151.84	132.23	12.9
s15850	597	759.44	537.11	29.3
Average				19.6

**Table 3: Current swing reduction.**

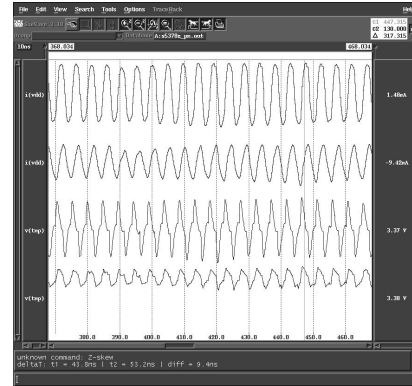
Circuit	Current Swing (mA)		
	Before optimization	After optimization	% reduction
s208	1.55	1.54	0.6
s420	3.00	2.68	11.9
s1423	15.69	4.2	73.2
s5378	30.69	15.37	49.9
s9234	17.88	10.1	43.5
s15850	14	6.54	53.3
Average			38.7

## 7. 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. 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] D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, G. Hoepfner, J. Kowaleski, K. Kuchler, M. Ladd, M. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala, and S. Santhanam. A 200MHz 64 b dual-issue CMOS microprocessor. In *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pages 1555–1566, 1992.

**Table 4: Voltage swing reduction.**

Circuit	Voltage Swing (V)		
	Before optimization	After optimization	% reduction
s208	0.12	0.11	8.3
s420	0.13	0.10	23.2
s1423	0.42	0.14	66.7
s5378	0.57	0.21	63.2
s9234	0.34	0.15	33.6
s15850	0.66	0.07	89.4
Average			47.4



**Figure 9: PowerMill simulation results of benchmark circuit s5378.**

- [5] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, 39(7):945–951, July 1990.
- [6] 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.
- [7] J. L. Neves and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Proc. Design Automation Conf*, pages 623–628, 1996.
- [8] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. checkTc and minTc: Timing verification and optimal clocking of synchronous digital circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 552–555, 1990.
- [9] 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.
- [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] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. *Journal of VLSI Signal Processing Systems*, 16(2/3):163–170, 1997.
- [16] Shiyu 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.