Optimization of Power Dissipation and Skew Sensitivity in Clock Buffer Synthesis

Jae W. Chung[†], De-Yu Kao[‡], Chung-Kuan Cheng[†], and Ting-Ting Lin[‡]

[†]Department of Computer Science and Engineering
Mail Code 0114
University of California, San Diego
La Jolla, California 92093-0114

[‡]Department of Electrical and Computer Engineering
Mail Code 0407
University of California, San Diego
La Jolla, California 92093-0407

Abstract

Contrary to a common belief that the load capacitance dominates overall power dissipation in the clock net, our experiment found a significant portion (up to 77.4%) of the power dissipation contributed from the interconnect of the clock tree. We introduce a new design concept and an algorithm to optimize both power dissipation and skew sensitivity in the clock buffer synthesis. Using a frequency divider and doublers, we effectively reduce the power dissipated in the clock tree into half. Our efficient algorithm optimizes power dissipation and clock skew sensitivity simultaneously. Our experimental results show an average of 49% reduction of power dissipation while reducing clock skew by several orders of magnitude.

1 Introduction

Operating frequency, power dissipation, and skew sensitivity are conflicting criteria for designing high performance synchronous VLSI systems. Power consumed by the clock net contributes a major portion of the total power consumption. Controlling the skew becomes harder as circuits get faster, chips become larger, and minimum feature size is scaled down. Clock buffer insertion and wire sizing to reduce *skew sensitivity* [7] will add power consumed by the interconnect of the clock tree.

A number of researches have been done to minimize the clock skew. The H-tree [1] approach can reduce clock skew successfully when the sinks are equal in size and are placed in a symmetric array. Jackson *et al*. [10] proposed the MMM (Method of Means and Medians) which is a generalization of the H-tree approach. Kahng *et al*. [11] then proposed a recursive geometric matching algorithm to reduce the wire length. Both [10] and [11] rely on linear delay model and focus on wire length balancing.

Tsay [14] devised a zero-skew clock routing scheme using the Elmore delay model. However, optimization of total wire length was not addressed in Tsay's work. Chao *et al*. [3] proposed a two-phase algorithm that reduces the wire length. Boese and Kahng [2] developed a DME (Deferred Merge Embedding) algorithm which is in principle identical to the second phase of [3]. Edahiro [8] proposed a linear algorithm in zero skew clock topology generation.

As we have noticed, no manufacturing parameters have been considered in these works. Thus, *zero skew* suffers from a considerable skew in practice. Also, all the previous works have been done under the assumption that a single clock source drives all the sinks distributed in the whole chip. As a result, excessive delays are introduced by the long interconnect wires between the clock source and the clock sinks. Recently Chung and Cheng [7] proposed a method of reducing the skew sensitivity and phase delay by inserting buffers. They considered wire width variations to achieve a reliable clock signal and proposed an efficient algorithm by adopting a dynamic programming method in finding optimal buffer locations in the clock tree. Lin *et al*. [12] presented a process variation tolerant clock routing mechanism.

Cong and Koh [5] proposed a simultaneous driver and wire sizing algorithm for delay and power dissipation minimization. They considered both capacitive power dissipation and short-circuit power dissipation. Hoppe *et al*. [9] describes a method of optimizing signal delay, chip area, and power dissipation using the transistor sizing approach.

However, no previous work addressed the problem of optimizing the power dissipation and skew sensitivity simultaneously. The insertion of clock buffers and wire sizing to reduce skew sensitivity will increase the power consumed by the interconnect. Thus the save of the power dissipation at the interconnect contributes a significant portion to the total power dissipation. Based on this observation, given a topology of clock tree and a library of buffers, we propose a new design concept as well as a power dissipation and skew sensitivity optimization algorithm.

The remainder of the paper is organized as follows. We formulate the power dissipation and skew sensitivity optimization problem in the next section. In section 3, we introduce the cost function and describe a wire width optimization for the power dissipation and skew sensitivity. We present a power dissipation and skew sensitivity optimization (PASSO) algorithm in section 4 along with the justification of applying dynamic programming approach. Section 5 contains our experimental results and comparisons.

2 Problem Formulation

In optimizing power dissipation and clock skew, we need to consider each contributing factor. *Supply voltage*, *operating frequency*, and *load capacitance* are major factors that contribute to the power dissipation. The motive behind the frequency divider/doubler is to reduce the *operating frequency* to reduce the power dissipation while guaranteeing the operation of the circuitry. We place the clock frequency divider at the root of the clock tree. Thus the clock frequency is reduced into half until the clock reaches a clock doubling circuit, which restores the original frequency.

Figure 1 describes a situation where the frequency divider is placed at the root and the doubler is placed at the level 2 of the clock tree. Here the clock frequency is reduced from the root level to level 2 and thus the power dissipation in this region of the clock tree is reduced into half. By intuition we can put the clock doubler at the bottom level of the clock tree for a maximal power save zone. However as the frequency doubler itself consumes power, putting a large number of frequency doublers may increase overall power dissipation. Besides, frequency doublers may introduce skew as well.

Hence, the power dissipation and skew sensitivity problem requires an optimal solution.

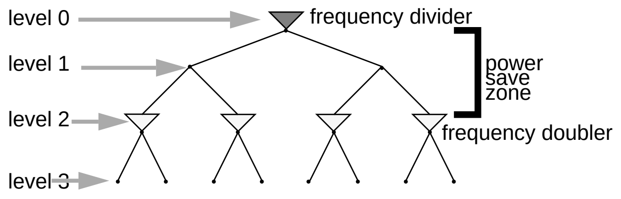

Figure 1: Power save zone with frequency divider/doubler

2.1 Power model

There are three components that establish the amount of power dissipation in a CMOS circuit [15]. These are:

1. Static dissipation (P_s) due to leakage current or other current drawn continuously from the power supply.

$$P_s = \sum_1^n I_{leakage} \cdot V_{DD} , \tag{1}$$

where n is the number of devices and V_{DD} is the supply voltage.

2. Dynamic dissipation (P_d) due to switching transient current or charging and discharging of load capacitances.

$$P_d = C_L \cdot V_{DD}^2 \cdot f_p , \tag{2}$$

where C_L is the total load capacitance, V_{DD} is the supply voltage, and f_p is the operating frequency.

3. Short-circuit dissipation (P_{sc})

$$P_{sc} = I_{mean} \cdot V_{DD} \tag{3}$$

Thus, the total power dissipation, P_{total} is the summation of three dissipation components.

$$P_{total} = P_s + P_d + P_{sc} \tag{4}$$

Among the three components, the dynamic power dissipation is the largest and may be used to estimate the total consumption of the circuit. The short circuit power dissipation can be avoided by careful circuit design, and the static dissipation is small compared to the dynamic power dissipation term. Thus throughout our work, we use the dynamic power dissipation as the total power dissipation using equation (2).

2.2 Frequency divider and frequency doubler

To save power and fulfill the timing requirements, the divider and doubler circuits should be as small and as fast as possible. The designs are shown in Figure 2 and 4.

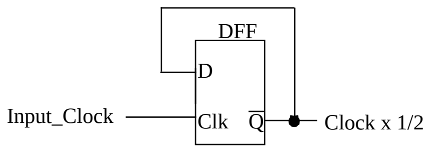

Figure 2: Frequency divider circuit

Assuming a 100 MHz input clock with 5 V V_{DD}, the power consumption according to above power model is 0.13 mW for each frequency divider circuit.

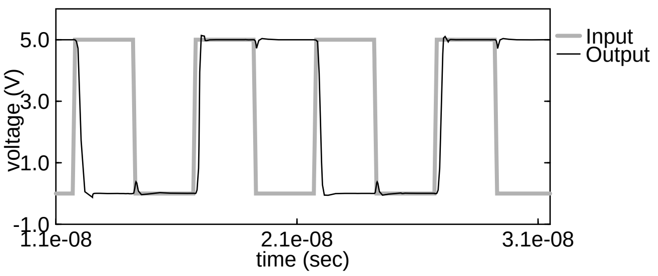

Figure 3: Divider simulation waveforms

Similarly, when we assume a 50 MHz input clock with a V_{DD} of 5 V, the power consumption of each frequency doubling circuit is 0.15 mW.

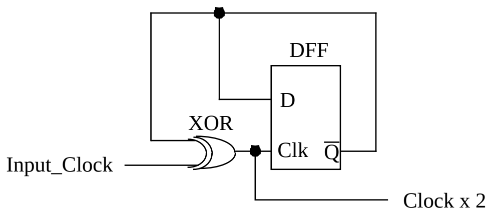

Figure 4: Frequency doubler circuit

Based on 0.5 micron CMOS technology, the operating frequency of divider and doubler circuit can run up to 300 MHz. The slope of the rising and falling edge can be modified by tuning the size ratio of the P-N transistors. The driving capacity of these two circuits is dependent on the transistor width. 36 transistors are used for a doubler and 20 transistors for a divider.

In Figure 6 we show the waveforms when the divider and doubler are cascaded together. Note how closely the doubler output follows the divider input.

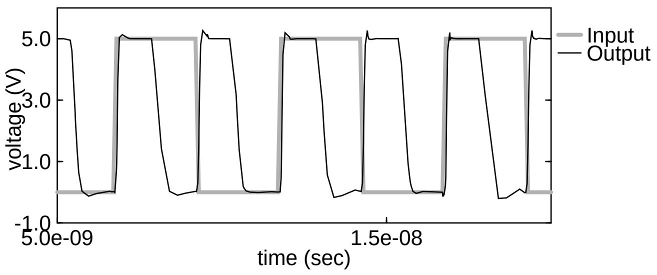

Figure 5: Doubler simulation waveforms

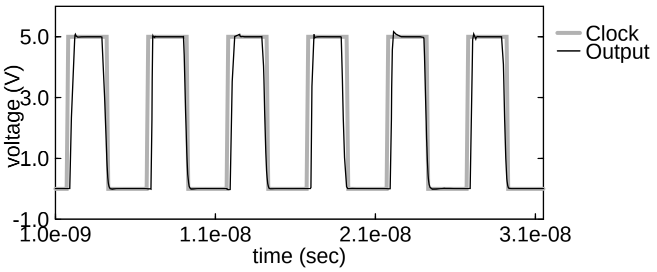

Figure 6: Waveforms of cascaded divider/doubler

2.3 Buffered clock model

Bakoglu[1] explains the effect of linear delay with interconnect length by inserting buffers in the interconnect. Using the lumped π model as a base timing model, we can model the clock buffer and interconnect wires as an equivalent circuit in Figure 7. Note that our buffer model includes frequency divider and frequency doubler.

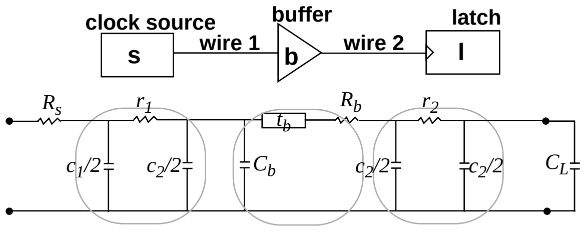

Figure 7: Buffered clock model

The clock source drives the buffer and the buffer drives the latch. C_L is the load capacitance of the latch and r_1, c_1, r_2, c_2 are resistance and capacitance values for interconnect wire 1 and wire 2.

Our timing model of inserting the clock buffer in the clock tree can be summarized as follows:

- Equivalent total subtree capacitance seen at buffer input is C_b, input capacitance of the clock buffer. This is an important assumption that isolates the buffered subtree from the whole system. We will make use of this *isolation property* of the buffer in applying dynamic programming algorithm.
- Internal delay of clock buffer, t_b, contributes to the total delay. Thus the delay from the clock source to the load, $t_{sl} = t_{sb} + t_{bl}$. Here, t_{sb} is the delay from the clock source to the buffer and t_{bl} is the delay from buffer to the load. $t_{bl} = t_b + t_l$.

2.4 Skew sensitivity

There are a number of circuit parameters that contribute to the clock skew. Some can be controlled by designers and some are under little designer control. Parameters such as process, voltage, and temperature variations are under least control by designers. Among these circuit parameters, we want to consider only those parameters over which designers have some control. We identify *wire width variations* and *buffer size variations* as parameters designers can control.

Given wire width, *w*, and buffer size, *s*, as control parameters, we define the skew sensitivity, *SS*, as the maximum difference between skews under varying values of *w* and *s*. Hence the goal of skew sensitivity minimization is to find the optimum *w* and *s* that achieve minimum *S*S under varying parameter variations. When we represent the wire width variations as w_δ, the skew sensitivity, *SS*, can be represented as the equation (5).

$$SS = \left| delay\left(w + w_\delta\right) - delay\left(w - w_\delta\right)\right| \approx \left|\frac{\partial}{\partial w} delay\right| \cdot 2w_\delta \quad (5)$$

Intuitively, simply increasing *w* and *s* seems to reduce the skew sensitivity. It is true to a certain point and the skew sensitivity decreases with larger *w* and *s*. However as the absolute value of propagation delay increases with larger *w* and *s*, their differences also increase and may result in larger skews. Increasing *w* and *s* also increases the power dissipation, which requires an optimal solution.

3 Power Dissipation and Skew Sensitivity Optimization

We can formulate the power dissipation and skew sensitivity optimization (PASSO) problem formally as follows.

Problem PASSO: Given a clock tree *T(E,V)* with *n* leaf nodes and a library of clock buffers with *s* different size clock buffers and frequency a divider/doublers, find an optimal level of frequency doublers and clock buffers with proper sizes and wire widths that optimizes the power dissipation and skew sensitivity.

3.1 Cost function

To present our power dissipation and skew sensitivity optimization (PASSO) algorithm, we define the cost function that reflect our objectives. The objective function of PASSO algorithm is the power dissipation and sensitivity of the clock skew under manufacturing variations. We maintain a matrix C[*b*, *l*, *s*, t], which represents the minimum cost with *b* buffer levels, the highest buffer locating at level *l*, with buffer size *s*, and location of the frequency doubler at *t*. The cost function is a weighted sum of power dissipation and skew sensitivity for each configuration. We introduce an optimization cost coefficient α which denotes how much weight we put on the skew sensitivity and power dissipation. Using α, the optimization cost coefficient, we define the cost as follows.

$$\mathrm{cos}t = (1 - \alpha) \cdot P + \alpha \cdot k_s \cdot SS \quad (6)$$

where *P* is the power dissipation, *SS* is the skew sensitivity, and k_s is the constant to normalize power dissipation and skew sensitivity so that their relative weight becomes equal.

Various optimization goals can be achieved by changing the value of α, the optimization cost coefficient. If we put α=1, then we are only interested in achieving skew sensitivity optimization. If we put α=0, we can optimize the power dissipation only. Depending upon the design requirements, we adjust the value of optimization cost coefficient α.

3.2 Wire width optimization

We can derive an equation of the cost as a function of *wire width*. In Figure 8 we calculate the propagation delay between two points *S* and *L*. The clock driver is driven by a source resistance of R_s and the load has capacitance C_L. The wire between two points have resistance value of r_w and c_w and modeled by π equivalent circuit.

Let *w* be the width and *l* be the length of the wire section between *S* and *L*. Also let k_r, k_c, k_p, and w_δ be the resistance coefficient, capacitance coefficient, power capacitance coefficient, and wire width variation respectively. When we assume the fringe capacitance effect is small, then $c_w = k_c \cdot w \cdot l$, $r_w = k_r \cdot l / w$, and $c_p = k_p \cdot w \cdot l$. If we let $a = (1-\alpha) \cdot k_p \cdot l$, $b = k_c \cdot R_s \cdot l$, $c = 1/2 \cdot k_r \cdot C_L \cdot l$, $d = \alpha \cdot k_s \cdot k_r \cdot w_\delta \cdot C_L \cdot l$, $k_d = 1/2 \cdot k_c \cdot k_r \cdot l^2$, $k = 2\alpha \cdot k_s \cdot k_c \cdot w_\delta \cdot R_s \cdot l$, equation (7) is the delay formula as a function of wire width between wire section *S* and *L*.

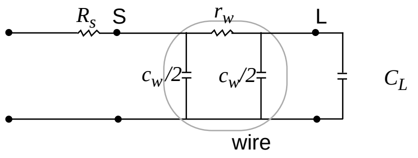

Figure 8: Cost optimization under wire width variations

Since power *P=aw* and from equation (5), by setting ∂cost/∂*w* equal to zero we get equation (9), which gives us the optimum wire width, w_{opt}, for the cost under wire width variations.

$$dealy_{SL} = b \cdot w + \frac{c}{w} + k_d \tag{7}$$

$$\frac{\partial}{\partial w}\mathrm{cost} = \frac{\partial}{\partial w}\left[(1-\alpha) \cdot P + \alpha \cdot SS\right] = \frac{\partial}{\partial w}\left[a \cdot w - \frac{d}{w^2} + k\right] \tag{8}$$

$$w_{opt} = min\left(\left[\frac{2d}{a}\right]^{1/3}, w_s\right) = min\left(\left[\frac{2\alpha \cdot k_s \cdot k_r \cdot w_\delta \cdot C_L}{(1-\alpha) \cdot k_p}\right]^{1/3}, \sqrt{\frac{k_r \cdot \left(C_L + k_f \cdot l/2\right)}{k_c \cdot R_s}}\right) \tag{9}$$

,where w_s is the optimal wire width for the skew sensitivity minimization derived in [7]. As α becomes close to 1, we emphasize skew sensitivity and choose w_s as an optimal wire width. In practice we limit the value of w_{opt} to be within the minimum and maximum limit.

Observation of equation (9) indicates that the optimum wire width is a function of α and load capacitance along with k_s and k_r, and does not depend on the length of the wire section between the source and load. As α gets smaller to optimize power dissipation, we get smaller wire width and as α gets larger to optimize skew sensitivity, wire width becomes wider. And as we choose larger w_δ, the wire width variation factor, we get wider wire width. Also according to equation (9), because load capacitance is larger at the higher level of the clock tree, wire widths at the top level is wider and becomes narrower as the level goes down the clock tree. This is desirable since clock skew is greatly affected by the width variation at the top level.

4 Power Dissipation and Skew Sensitivity Optimization (PASSO) algorithm

Based on the optimization techniques presented in section 3, an algorithm to synthesize an optimal clock buffer tree is presented. We assume the same size buffer on the same level of the tree. Also, the proposed algorithm searches buffers of finite sizes available in the given library. Thus we fix the parameters of the minimum size buffer and assume an integer multiples of buffer sizes. We start our algorithm from the given topology. This topology can be generated by any clock topology generation program. We used SASPO [4] algorithm, which is known to give the best result.

4.1 PASSO algorithm

In Figure 10 we present our power dissipation and skew sensitivity optimization (PASSO) algorithm. We maintain a matrix *C*[*b, l, s,* t], which represents the minimum cost with *b* buffer levels, the highest buffer locating at level *l*, with buffer size *s*, and location of the frequency doubler at *t*. This matrix is used as a lookup table in the dynamic programming algorithm. The algorithm starts by extracting the clock topology from the given circuit. The extracted clock topology, *ClockTree*, by *ReadTopology*() is used throughout the algorithm. *BufferInsert*() is the main function which inserts buffers of optimal sizes to minimize the cost function. After the completion of *BufferInsert()*, the content of lookup table *C*[*b, l, s, t*] will be complete and we can retrieve all the buffer parameters which yield the minimum cost function.

BufferInsert() calls *ConstructTable*(*b, l, s, t*) to build the lookup table *C*[*b, l, s, t*] for all *b, l,* s, and *t*. For each iteration, the algorithm reads the lookup table and checks to see if there is a combination of buffer location *l*, with buffer size *s*, and frequency doubler at *t* that yields smaller cost function. The necessary check can be defined based on the value of *t*, the location of the frequency doubler.

The possible value of *t* are:

i) *t* = 0, if frequency doubler is located above *l*, the location of the highest buffer.

$$C[b, l, s, 0] = \min_{1 \le l' < l,\ 1 \le s' \le s}\left(calcCost(l, s, l', s', 0) + C[b-1, l', s', 0]\right) \tag{10}$$

ii) *t* = 1, if frequency doubler is located at *l*, the location of the highest buffer.

$$C[b, l, s, 1] = \min_{1 \le l' < l,\ 1 \le s' \le s}\left(calcCost(l, s, l', s', 1) + C[b-1, l', s', 0]\right) \tag{11}$$

iii) *t* = 2, if frequency doubler is located below *l*, the location of the highest buffer.

$$C[b, l, s, 2] = \min_{1 \le l' < l,\ 1 \le s' \le s}\left(\left(calcCost(l, s, l', s', 2) + C[b-1, l', s', 2]\right), \left(calcCost(l, s, l', s', 3) + C[b-1, l', s', 1]\right)\right) \tag{12}$$

,where we make use of the principle of optimal subproblems to compute the skew of the higher level buffer.

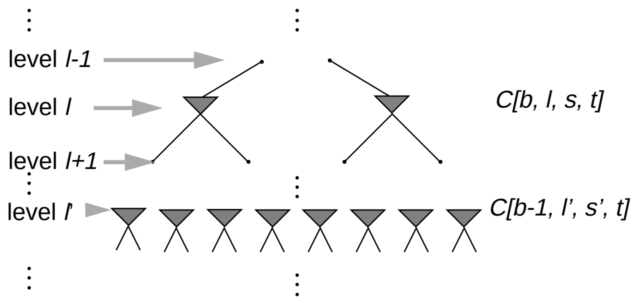

Figure 9: Calculation of $C[b, l, s, t]$

To compute $C[b, l, s, t]$ we need to know the values of *calcCost*(l, s, l', s', t) and $C[b\text{-}1, l', s', t]$. These values are obtained by *calcCost*() procedure and from the lookup table C[], respectively. Figure 9 illustrates the calculation of $C[b, l, s, t]$. While calculating the partial cost, the algorithm performs the *WireSizing*() for optimal wire size between level l and level l' as outlined in section 3. *calcCost*() then selects two paths in the clock tree and returns the calculated partial cost from level l to level l'. Two paths *path 1* and *path 2* are set by varying uniform wire widths by w_δ and by varying fixed buffer size by s_δ on two different paths in the given clock tree.

```
Algorithm PASSO
Choose $\alpha$, the optimization coefficient;
ClockTree = ReadTopology();
BufferInsert(ClockTree, BufferLibrary);
Select the best result;
print minimum cost, buffer levels, and buffer sizes;

Procedure BufferInsert(ClockTree, BufferLibrary)
for $b$ := 1 to maxbuffer do
    for $l$ := 1 to maxlevel do
        for $s$ := 1 to maxsize do
            C[$b, l, s, t$] = ConstructTable($b, l, s, t$);
return C[$b, l, s, t$];

Procedure ConstructTable($b, l, s, t$)
for $l'$ := 1 to $l$-1 do
    for $s'$ := 1 to $s$ do
        cost = calcCost($l, s, l', s', t$) + C[$b$-1, $l', s', t$];
choose minimum cost;
return cost;

Procedure calcCost($l, s, l', s', t$)
insert buffer at $l$ with size $s$;
insert buffers at $l'$ with size $s'$;
WireSize($l, l'$);// do wire sizing between level $l$ and $l'$
(path, path2) = SetPath($l, l'$);// setting of path1 and path2
partial cost = cost(path1, path2, $l, l'$);// partial cost
return partial cost;
```

Figure 10: PASSO algorithm

4.2 Run time analysis

The run time of our skew sensitivity minimization algorithm is dominated by for loops. Since there are two loops for the number of buffers and the buffer levels in procedure BufferInsert(), it takes $O(log^2n)$ time. Two loops for different buffer sizes in *BufferInsert*() and *ConstructTable*() require $O(s^2)$. Wire sizing, buffer sizing, and path selection operation in *calcSkew*() require $O(logn)$ time. Together, the total run time is $O(log^3n\cdot s^2)$ and it is polynomial. Algorithm PASSO uses the cost table and uses $O(log^3n)$ extra space. A recursive solution would save the space, however, the investment of the extra space to produce faster algorithm is worthwhile.

We now show that the complexity of an exhaustive approach is computationally expensive. Since n is the number of leaf nodes, $logn$ is the number of levels in $T(E,V)$. Trying all different buffer sizes, including no buffer, for each level requires $O((s+1)^{logn})$ time. This can be rewritten as $O(n^{log(s+1)})$, and this is a polynomial time complexity with high exponent. Note, however, that when we consider *logn* as an input (number of levels in the clock tree), algorithm PASSO remains polynomial and the run time of an exhaustive method becomes exponential.

5 Experimental results

Proposed algorithm, power dissipation and skew sensitivity optimization (PASSO), is implemented and tested on five benchmark circuits. The buffer parameters are derived from [13] and based on the 0.5 micron CMOS technology. The output resistance is 3170 Ω, input capacitance is 10 fF, and the buffer intrinsic delay is 35.5 ps for 1X buffer. Wire capacitance values are based on five benchmark data (*r1* through *r5*) and adjusted to 0.5 micron CMOS technology. We used K_r of 0.12 Ω/micron and K_c of 0.05 fF/micron. The frequency doubler and divider parameters are obtained from Spice simulations. For the frequency doubler, the output resistance is 710 Ω, input capacitance is 51 fF, and the intrinsic delay is 153 ps. For the frequency divider, the output resistance is 1220 Ω, input capacitance is 33 fF, and the intrinsic delay is 52 ps. We used a value of w_δ, wire width variation factor, to be 15% of the unit wire width (0.075 micron for 0.5 micron technology) and s_δ to be 15% of the unit buffer size to simulate the worst case manufacturing variations. The skew sensitivity of the frequency divider is 0.98 fF and 17.41 fF for the frequency doubler. Table 1 compares the load capacitance values with the interconnect capacitance.

Table 1: Interconnect capacitance vs. total capacitance

benchmarks (# of pins)	**load capacitance (farad)**	**interconnect capacitance (farad)**	**interconnect capacitance/ total capacitance**
r1 (267)	$6.09\cdot10^{-12}$	$2.08\cdot10^{-11}$	77.4 %
r2 (598)	$1.47\cdot10^{-11}$	$4.54\cdot10^{-11}$	75.5 %
r3 (862)	$3.87\cdot10^{-10}$	$6.47\cdot10^{-11}$	62.6 %
r4 (1903)	$9.67\cdot10^{-11}$	$1.41\cdot10^{-10}$	59.3 %
r5 (3101)	$1.20\cdot10^{-10}$	$3.82\cdot10^{-10}$	76.1 %

As shown in the last column of Table 1, we found the interconnect capacitance dominates and the percentage of interconnect capacitance to the total capacitance ranges from 59% to 77%. This values are measured before wire sizing is performed and before clock buffers are inserted. Interconnect capacitance will increase if wire sizing is performed and clock buffers are added. Thus the power save in the interconnect can result in a significant save of overall power, which is shown in Table 2.

Table 2: Power and skew values for α=0

bench marks	power before PASSO (farad)	power after PASSO (farad)	skew before PASSO (sec)	skew after PASSO (sec)
r1	$2.08 \cdot 10^{-11}$	$1.08 \cdot 10^{-11}$	$4.38 \cdot 10^{-10}$	$4.80 \cdot 10^{-11}$
r2	$4.54 \cdot 10^{-11}$	$2.29 \cdot 10^{-11}$	$4.83 \cdot 10^{-10}$	$6.21 \cdot 10^{-11}$
r3	$6.47 \cdot 10^{-11}$	$3.28 \cdot 10^{-11}$	$1.52 \cdot 10^{-09}$	$1.54 \cdot 10^{-10}$
r4	$1.41 \cdot 10^{-10}$	$7.22 \cdot 10^{-11}$	$5.40 \cdot 10^{-09}$	$2.73 \cdot 10^{-10}$
r5	$3.82 \cdot 10^{-10}$	$1.96 \cdot 10^{-10}$	$6.81 \cdot 10^{-09}$	$1.47 \cdot 10^{-9}$

Table 2 lists the power dissipation and skew values for α=0, where we want to optimize power dissipation only. The power dissipation values are represented in *equivalent capacitance* discussed in section 2.2. Compared to the power dissipation before we apply the PASSO algorithm, we achieve an average of 49% reduction in power dissipation. Please note we still achieve an order of reduction in clock skew at the same time.

Table 3 lists the power dissipation and skew values for α=1, where we want to optimize clock skew only. Compared to the clock skew before we apply PASSO algorithm, we achieve several orders of reduction in clock skew values. Also note we still achieve about 37.5 % reduction in power dissipation at the same time.

Table 3: Power and skew values for α=1

bench marks	power before PASSO (farad)	power after PASSO (farad)	skew before PASSO (sec)	skew after PASSO (sec)
r1	$2.08 \cdot 10^{-11}$	$1.31 \cdot 10^{-11}$	$4.38 \cdot 10^{-10}$	$4.77 \cdot 10^{-15}$
r2	$4.54 \cdot 10^{-11}$	$2.93 \cdot 10^{-11}$	$4.83 \cdot 10^{-10}$	$7.46 \cdot 10^{-14}$
r3	$6.47 \cdot 10^{-11}$	$3.44 \cdot 10^{-11}$	$1.52 \cdot 10^{-09}$	$9.62 \cdot 10^{-14}$
r4	$1.41 \cdot 10^{-10}$	$8.52 \cdot 10^{-10}$	$5.40 \cdot 10^{-09}$	$1.43 \cdot 10^{-13}$
r5	$3.82 \cdot 10^{-10}$	$2.73 \cdot 10^{-10}$	$6.81 \cdot 10^{-09}$	$2.51 \cdot 10^{-13}$

Normally designers want to choose the value of optimization cost coefficient α somewhere between 0 and 1 depending upon the goal of optimization. Thus we list the relationship between the power dissipation and skew as a function of α, an optimization cost coefficient in Figure 12. We observe the decrease of the skew sensitivity as α approaches 0. As α approaches 1, we see the reduction of power dissipation.

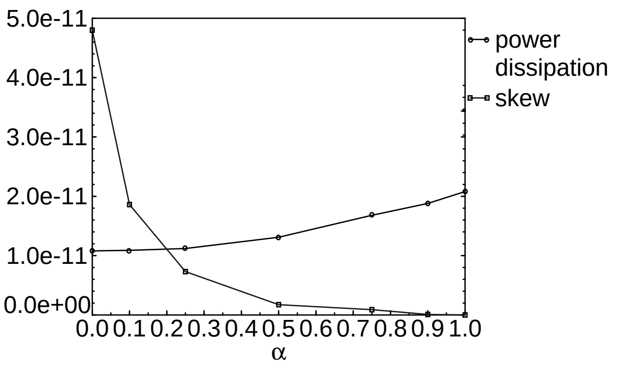

Figure 11: Power dissipation and skew sensitivity as a function of α

As we observe in Figure 11, skew sensitivity values are more sensitive to α than the power dissipation. It shows the rapid decrease of clock skew and relatively slower increase of power dissipation as α increases when α is smaller than 0.5. Thus the proper value of α may be obtained at the ***knee***, where the clock skew starts to decrease more slowly. In Figure 11, a good ***knee*** point can be somewhere between 0.25 and 0.5. By letting designers to choose α between 0 and 1, we can achieve optimization of power dissipation and skew sensitivity simultaneously.

References

[1] H. Bakoglu, "Circuits, Interconnections and Packaging for VLSI," Addison-Wesley, 1990.

[2] K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees with Minimum Wire-length," Proc. 5th IEEE Intl. Conf on ASIC, NY, pp. 17 - 21, 1992.

[3] T.-H. Chao, Y.-C. Hsu, and J.-M. Ho, "Zero-Skew Clock Net Routing," Proc. 29th ACM/IEEE Design Automation Conf., pp. 518-523, 1992.

[4] N.-C. Chou and C.-K. Cheng, "Wire Length and Delay Minimization in General Clock Net Routing," Proc. IEEE Intl. Conf. on Computer-Aided Design, pp. 552-555, 1993.

[5] J. Cong and C.-K. Koh, "Simultaneous Driver and Wire Sizing for Performance and Power Optimization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Dec. 1994, vol. 2, (no. 4), pp. 408-425.

[6] T. Cormen, C. Leiserson, and R. Rivest, "Introduction to Algorithms," MIT Press, 1990.

[7] J. Chung and C.-K. Cheng, "Skew Sensitivity Minimization of Buffered Clock Tree," Proc. IEEE Intl. Conf. on Computer Aided Design, pp. 280-283, 1994.

[8] M. Edahiro, "An Efficient Zero-Skew Routing Algorithm," Proc. 31st ACM/IEEE Design Automation Conf., pp. 375-380, 1994.

[9] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, and W. Specks, "Optimization of High-Speed CMOS Logic Circuits with Analytical Models for Signal Delay, Chip Area, and Dynamic Power Dissipation," IEEE Transactions on Computer Aided Design, vol 9. no. 3., March 1990.

[10] M. A. B. Jackson, A. Srinivan, and E. S. Kuh. "Clock Routing for high performance ics." Proc. Design Automation Conferences, pp. 573-579, 1990

[11] A. Kahng, J. Cong, and G. Robins. "High-Performance Clock Routing Based on Recursive Geometric Matching," Proceedings of Design Automation Conferences, pp. 322-327, 1991

[12] S. Lin and C. K. Wong, "Process-Variant-Tolerant Clock Skew Minimization," Proc. IEEE Intl. Conf. on Computer-Aided Design, pp. 284-288, 1994.

[13] T. Sakurai, "A Unified Theory for Mixed CMOS/ BiCMOS Buffer Optimization," IEEE Journal of Solid-State Circuits, vol. 27, no. 7, July 1992.

[14] R.-S. Tsay, "Exact Zero Skew," Proc. IEEE Intl. Conf. on Computer Aided Design, pp. 336-339, 1991.

[15] N. Weste and K. Eshraghian. "Principles of CMOS VLSI Design," Addison Wesley, pp. 231-237, 1991