A Practical Clock Router that Accounts for the Capacitance Derived from Parallel and Cross Segments

Mitsuho Seki, Syun'ichi Kobayashi, Kazuo Kato, Koki Tsurusaki
Semiconductor Integrated Circuit Division, Hitachi Ltd., Tokyo, 185 Japan
email : m-seki@crl.hitachi.co.jp

Abstract

We propose a practical clock router that takes the capacitance caused by parallel and cross segments into account. In a conventional clock router that doesn't consider this capacitance, clock skew calculated by SPICE after layout was 220-650 ps, which was 20-100 times greater than the reported skew of the router. In our router, skew in SPICE was reduced to 20-240 ps and only 2-4 times greater than our router's report.

1. Introduction

Reducing cycle time is essential to achieve high performance LSIs. However, if clock skew isn't minimized along with a frequency increase, cycle time may not be reduced. Therefore, minimizing clock skew is important.

Clock routing methods that find a delay balance point between two input terminals (Tsay[1],Chao[2]) are often used. Pullela[3] and other extensions of [1][2] are also known. Furthermore, clock phase delay minimization methods (Pullela[4], Edahiro[5], Cheng[6]) have been eagerly studied. On the other hand, we have proposed in Seki[7] a clock routing method that minimizes clock skew by realizing a specified delay on each clock net.

However, these conventional clock routers only consider segment capacitance proportional to the segment length. They don't consider the capacitance in two- or three- dimensional terms. In deep submicron technology, the distance between two parallel segments is very narrow and there is a high density of signal segments. The influence of these capacitances on signal delay is likely to become serious. In fact, our experiments in 0.8 μ m technology show that if we don't consider the parallel and cross capacitance derived from other segments, the value of clock skew calculated by SPICE after layout is more than 20, and sometimes 100, times larger than the reported value in our previous clock router (Seki[7]). Clearly, conventional clock routers designed without consideration of the capacitance derived from parallel and cross segments cannot be practical for the actual high speed LSI design. This surprising difference will become larger and larger as LSI technology progresses.

Because we cannot accurately predict in advance how much segments cross each other or run in parallel, clock routers that don't take the capacitance derived from parallel and cross segments into consideration will not be able to assure a minimal clock skew. Therefore, we must take the precise parallel and cross-segment capacitance into account in the clock routing. We call this capacitance derived from parallel and cross segments "PC-capacitance" in this paper.

Routing methods to reduce crosstalk by parallel segments (Chaudhary[8], Gao[9], Kirkpatrick[10]) have been proposed. However, these methods only minimize the total parallel segment length, and don't lead to clock skew minimization. Here, we extend a clock router described in Seki[7] and minimize clock skew by considering the PC-capacitance.

PC-capacitance may be estimated by global routing. But this is only an estimation, because, for example, a global routing cannot determine which two segments are really adjacent in the detail routing. So, we adopt the following strategy. First, we find an initial clock route as in Seki[7]. Second, we route all ordinary signals. Finally, our detail clock router is executed, which modifies the initial clock routes while considering the PC-capacitance. The PC-capacitance extraction model of the detail clock router is almost the same as commonly used load extraction tools.

In section 2, a clock routing strategy considering PC-capacitance is described. Our capacitance extraction model is introduced in section 3. A clock routing flow and a detail routing algorithm are explained in section 4. Finally, our experimental results are discussed in section 5.

2. Clock Routing Strategy

2.1 Assumptions and Net Delay Model

Our clock distribution logic is made before the placement and routing, and is appropriately partitioned into nets by inserting buffers as shown in Fig. 2.1. The position of buffers and flipflops are decided before the clock routing. Our clock routing is executed net by net over the entire clock distribution logic, and a clock net is assumed to consist of one output teminal and multiple input terminals. As mentioned in Seki[7], a specified delay from the output terminal to each input terminal is given. This delay is specified by reflecting the floorplan result. Generally, the same value is given to each input terminal in the case of clock nets. If placement cannot satisfy the specified delay, our clock router will try to route as close as possible to the specified delay while simultaneously minimizing the skew. We use the method of Rubinstein[11] based on Elmore[12] as a net delay calculation method. The capacitance and resistance model used for each segment is the π model. Capacitance between the bottom and side of a segment and substrate is

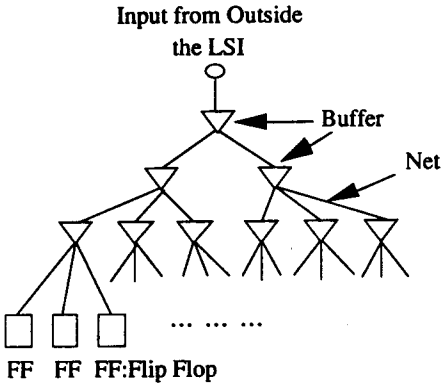


Fig. 2.1 Clock Distribution Logic

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00 © 1996 IEEE

proportional to the segment length. For a through-hole, resistance is considered but capacitance is ignored.

2.2 *Total Flow of Our Clock Routing*

Our clock routing taking the PC-capacitance into consideration is performed according to the flow shown in Fig. 2.2.

First, our initial clock router described in Seki[7] runs. At this time, we use an average PC-capacitance rate which is derived from many experiments. It's because we cannot realize the specified net delay unless we estimate, to some degree, how much the segment capacitance is increased by PC-capacitance. The delay adjustment to minimize the routing skew is done by U-shaped modification of a basic pattern consisting of one trunk and multiple branches. This method is explained later.

Next, we route ordinary signals (data signals etc.) under the condition that the initial clock routes are fixed. Finally, our detail clock router is executed while taking the actual PC-capacitance into consideration. The same delay adjustment strategy as in our initial clock router is followed in this detail clock routing to meet the specified delay and to minimize clock skew. However, there aren't many routing grids left for U-shaped modification because the ordinary signals are routed at a high density. Therefore, several small modifications per net are performed.

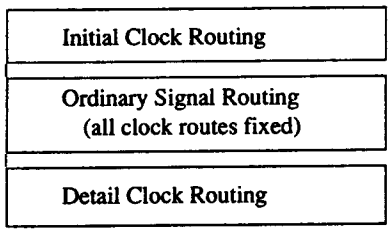


Fig. 2.2 Clock Routing Total Flow

2.3 *Importance of Pattern Modification*

It becomes more important to modify initial clock routes when considering PC-capacitance. For example, let us assume that we use a delay balance point clock router as our detail clock router. In the detail clock routing phase after ordinary signals are routed, calculation of a new tapping point location becomes very difficult because the occurence of PC-capacitance, approximately generates a condition where the capacitance of a segment isn't proportional to the segment length. Furthermore, the tapping point movement is very much restricted owing to the high-density of the ordinary signals routes. As a result, detours frequently have to be introduced. The goal of a detour must be to realize some target delay because a detour changes the resistance and capacitance of a route between a tapping point and an input terminal. A U-shaped modification is useful and practical for this purpose.

3. *Parallel and Cross Segments Capacitance Extraction Model*

In deep submicron technology, the capacitance derived from a voltage difference between two parallel segments becomes relatively large because the routing grid distance is so narrow. As for crossed segments, the capacitance of a single cross is very low, but there are numerous cross points between the segments because of the high-density routing. So, we can't ignore total cross capacitance, either. We execute our detail clock router while taking these two- or three- dimensional capacitances into consideration. Our clock router can calculate the segment capacitance as precisely as typical load extraction tools.

3.1 *Basic Model*

The basic capacitance extraction model is illustrated in Fig.3.1. Capacitance against the substrate usually consists of the following two factors. One is the capacitance Cs between the bottom of a segment and the substrate, the other is the fringe capacitance Cf between the side of a segment and the substrate. These two capacitances are proportional to the length of a segment.

The parallel segment capacitance Cp is derived from the area where two segments run in parallel on the same routing layer. We assumed that a designer will specify as a technical parameter how close segments should be parallel capacitance needs to be considered. This capacitance is also proportional to the length of a segment.

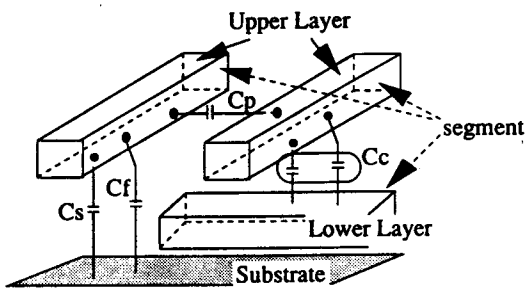


Fig. 3.1 Parallel & Cross Capacitance Model

The cross segment capacitance Cc is derived from an area where two segments on different layers cross each other. If many segments cross a focused segment, the cross capacitance of the focused segment is proportional to the total intersectioning area. Furthermore, as shown in Fig. 3.2, if two segments cross on different layers, there is cross capacitance at the crossing area between the two layers, but there isn't any capacitance against that part of the substrate because the upper layer segment is shielded from the substrate by the lower layer segment.

3.2 *Fat Wire Model*

In our routing model, the width of a fat wire, for example, a power segment, is an integer multiple of the routing grid distance. We decompose the fat wire into a group of segments by slicing it along the main running direction. For example, the fat wire on the 1st metal layer is sliced in the X direction and one of the sliced segments is dealt with as a normal segment. In this way, we can adapt the above

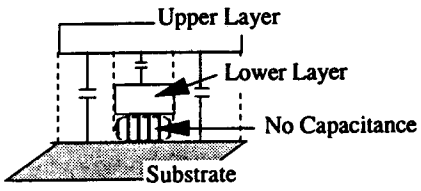


Fig. 3.2 Detail of Cross Capacitance Model

concept of PC-capacitance naturally.

The width of normal segments is defined so that they don't come into contact with each other when they are on the routing grid. On the other hand, the width of sliced segments from a fat wire is defined such that they just come into contact with each other. Either kind of width can be given to a segment in our clock routing.

4. Detail Clock Routing Algorithm

4.1 Initial Clock Routing Pattern

Here, we describe the outline of our initial clock routing as shown in Seki[7]. Our clock router aims at realizing a specified delay at each input terminal. To minimize clock skew for a multiphase clock, including a gated clock, the specified delay must be realized at each net or input terminal.

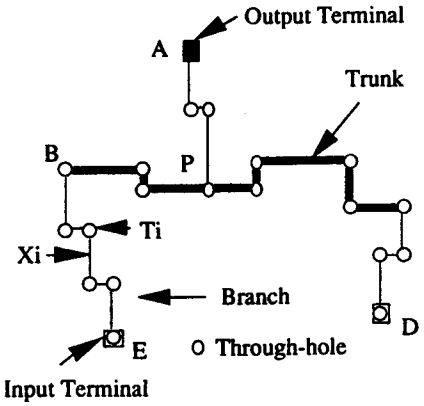
Because the net delay in Rubinstein[11] cannot be calculated until a routing pattern is determined, we cannot judge whether the specified delay can be realized or not in advance. So, we assume a basic routing pattern consisting of one trunk and multiple branches as illustrated in Fig. 4.1(b). Then, we modify this basic pattern by calculating the difference between the delay of the basic pattern and the specified delay. Though straight lines are assumed for the trunk and branches in the basic pattern, a route can be bent in an arbitrary place. For instance, the trunk and branches are bent in Fig.4.1(a).

First, a trunk is drawn. Next, a route from the trunk to an output terminal is created using multiple routing layers and through-holes. After that, we determine the diverging points on the trunk to route to each input terminal. Because the routing pattern is already decided so far if we assume a branch is a straight line, a net delay calculation is executed by defining the straight branch length as a variable. We set this net delay equal to the specified delay, and solve this equation to get each necessary branch length. Finally, from the necessary branch length obtained above, we derive the capacitance and resistance conditions that each branch should satisfy. Branches are modified until the capacitance and resistance conditions are satisfied. The modification is performed in a U-shape.

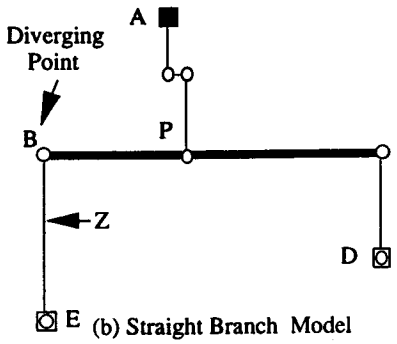
4.2 Detail Routing Flow

In Fig. 4.2, we show a flow of our detail routing in which our initial clock routing patterns are modified after ordinary signal routing.

Initially, all signals'routes including power segments are input. Then, we extract the capacitance of clock nets based on the PC-capacitance extraction model described in the previous section, and calculate the delay of the clock nets.



(a) Actual Routing Model



(b) Straight Branch Model

Fig. 4.1 Branch Routing Model

Next, we define a diverging point on the trunk(see Fig. 4.1). Then, we adjust the delay from P to each input terminal in each clock net considering the PC-capacitance by modifying the branches. Routing skew in a clock net can be eliminated by this process.

At this point, we calculate the delay of each clock net again because it might have been changed by the branch modification. Then, the internet delay adjustment is executed. This adjustment is done for a group of nets that have the same level from the starting point of the clock distribution logic. A route from an output terminal to a trunk (see Fig. 4.1(a), between A and P) is modified to adjust the internet delay. The route modification method is the same as that of the branch modification in the initial clock routing.

However, a net routing order causes a problem. The delay of a formerly processed net is changed by the PC-capacitance since the routes of a modified net run in parallel with or cross earlier processed nets. We can alleviate this problem by iterating our detail clock router a few times because the degree of modification gradually decreases with each iteration.

4.3 Calculation of the Branch Modification Amount

We adjust the delay of each branch to make the intranet skew equal to zero by modifying it. For this purpose, we have to calculate how much we detour by U-shape.

First, the delay of a branch (a route from P to each input terminal) with PC-capacitance is calculated. Let us assume that the branch BE consists of n segments and the length of

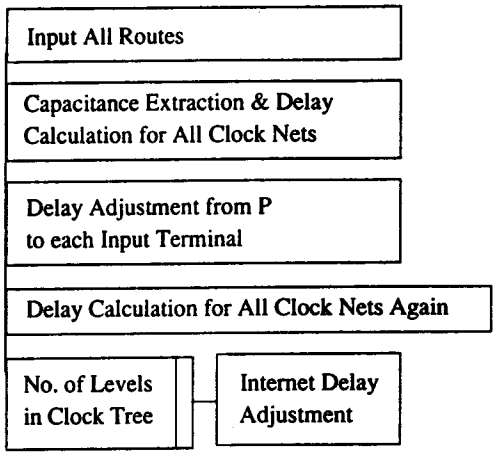


Fig. 4.2 Detail Clock Routing Flow

the i-th segment in this branch is X_i. We define the unit length capacitance composed of the bottom and fringe capacitance as c_j, the unit length resistance as r_i, and resistance of the i-th through-hole as T_i. Capacitance h_j is the PC-capacitance derived from other segments. I0 is the input capacitance of the input terminal which is connected with the branch. Let Y_k be the length of each segment from P to B, RF_k be the unit length resistance of these segments and UR be the total segment resistance between P and B. Then, PB_delay1, the delay of a branch from P to B is as follows.

$$PB\_delay1 = \sum_{i=1}^{n+1} T_i \left(\sum_{j=i}^{n} c_j \cdot X_j + h_j + I0\right) + \sum_{i=1}^{n} r_i \cdot X_i \cdot \left((c_i \cdot X_i + h_i)/2 + \sum_{j=i+1}^{n} (c_j \cdot X_j + h_j) + I0\right) + UR \cdot \left(\sum_{i=1}^{n} c_i \cdot X_i + h_i\right) \quad (4.1)$$

$$UR = \sum_{k=1}^{m} RF_k \cdot Y_k + \sum_{k=1}^{m} T_k \quad (4.2)$$

This PB_delay1 is calculated for each branch, and we set the maximum PB_delay1 from among all branches as max_PB_delay1.

When considering the delay from P to each input terminal, because modification of one branch influences the delay of the other branches, branch-by-branch sequential modification cannot adjust the intranet skew. Therefore, we have to calculate the necessary amounts of modification for all branches at the same time while considering delays to all input terminals. To do that, we transform PB_delay1 to the straight branch model delay, say PB_delay2, so that we can use the branch modification method in our initial clock routing. We calculate the amount of modification needed by each branch at a time by the method shown in Seki[7].

Let us define the straight branch segment length as Z, the unit length resistance as R, the unit length capacitance against substrate as C, and the PC-capacitance of other segments as H. Then,

$$PB\_delay2 = R \cdot Z \cdot ((C \cdot Z + H)/2 + I0)) + UR \cdot (C \cdot Z + IO + H) \quad (4.3)$$

We set PB_delay1 = PB_delay2 for each branch, and solve this quadratic equation for Z.

Next, the amount of branch modification is calculated to adjust routing skew inside a net. In our initial clock routing, the amount of branch modification for the one-trunk-plural-branch pattern needed to realize the specified delay is calculated. If the positive modification amount cannot be obtained, the specified delay cannot be realized. In this case, we gradually increase the specified delay until we can get the positive modification amount.

The specified delay to calculate the branch modification amount in our detail routing is begun from the max_PB_delay1 because this maximum delay is a minimum target delay that must be realized to adjust clock skew inside a clock net. This delay is increased little by little until a positive modification amount is determined. We call this increased delay, the "final target delay". Then, we modify a branch using a U-shaped pattern so that delay difference with an actual route becomes almost zero.

We have finished the delay adjustment of the branches for all nets by this stage. This means the intranet skew adjustment. Then, we try to adjust the internet delay. This is done by modifying a route from an output terminal to the trunk as follows.

To calculate the delay of the route from A to P in Fig. 4.1(a), the following notations are introduced. O0 is the output resistance, the length of the i-th segment from A to P is XF_i, the capacitance and the resistance corresponding to XF_i are cf_i and rf_i, respectively, hf_j is the PC-capacitance derived from other segments and TF_i is the resistance of the i-th through-hole. In this delay adjustment stage, because routes from P to each input terminal are fixed, we can express a sum of capacitance derived from these fixed routes and terminals as W. Then, the delay from A to P, say AP_delay, is as follows.

$$AP\_delay = \sum_{i=1}^{n} TF_i \left(\sum_{j=i}^{n+1} cf_j \cdot XF_j + hf_j + W\right) + \sum_{i=1}^{n} rf_i \cdot XF_i \cdot ((cf_i \cdot XF_i + hf_i)/2 + \sum_{j=i+1}^{n} (cf_j \cdot XF_j + hf_j) + W) + O0 \cdot \left(\sum_{i=1}^{n} cf_i \cdot XF_i + hf_i\right) \quad (4.4)$$

We calculate this AP_delay for nets having the same level in a clock distribution logic. Then, the delay between A and P for these nets is adjusted using a U-shaped pattern so that it can reach the maximum AP_delay among these nets.

4.4 Branch Modification Pattern

From the calculation in the section 4.3, we can obtain how much an initial clock pattern should be modified. Now,

it is time to find a U-shaped detour route as shown in Fig. 4.3. We use the line search method proposed by Kitazawa[13] to find all the routes in clock routing.

We choose the longest segment included in a branch so that we can prepare an appropriate number of U-shaped detour applicants even if there is a high density ordinary signal segments. But sometimes, there are many U-shape detour applicants in this segment. In such a case, we reduce the number of detour applicants.

We define the delay difference between the final target delay and the delay of the initial clock net pattern as D. As shown in Fig. 4.3, assume that there is a segment perpendicular to segment T in an arbitrary location. If we set the length of this segment as x, at least the capacitance proportional to 2x and the PC-capacitance of other segments are added to the capacitance of the current branch. We calculate the delay increment D1 by multiplying this incremental capacitance with the resistance of an output terminal side from segment T. Because four through-holes are introduced by the U-shaped modification, the actual delay increment due to the U-shaped modification must be larger than D1. So, if D1 is larger than D, we don't adopt the segment with a length greater than x.

Next, let us consider the length of segment S which is parallel to segment T. If we set the length of this segment as y and the length of the segments located on each side of segment S as x, at least the capacitance proportional to 2x+y and the PC-capacitance of the other segments are added to the capacitance of a current branch. When calculating the delay increment of the range just above x, we can get the range of length y.

Next, let us consider the length of segment S which is parallel to segment T. If we set the length of this segment as y and the length of the segments located on each side of segment S as x, at least the capacitance proportional to 2x+y and the PC-capacitance of the other segments are added to the capacitance of a current branch. When calculating the delay increment of the range just above x, we can get the range of length y.

We search among the U-shaped modification applicants using the method of Kitazawa[13] in the above range of x,y. If a single detour isn't enough to realize the final target delay, the same detour process is repeated on the next longest segment in the branch.

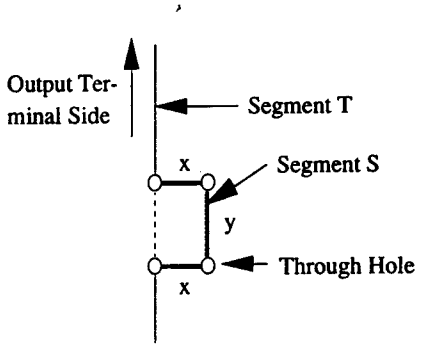


Fig. 4.3 U-Shaped Detour

5. Experimental Results

We implemented this algorithm on a SUN SparcStation10 in C language.

Experiments were done on gate arrays as follows. As the first step, our initial clock router was executed with the empirical value of an average PC-capacitance rate, then ordinary signals were routed as the second step. As the third step, we executed the following two cases. In the first case, we calculated the maximum clock path delay and clock skew with SPICE using the result of the second step. In the second case, our detail clock router that takes the PC-capacitance into consideration was executed, then SPICE was done using this result. Because our clock router uses the Rubinstein[11] delay calculation unlike SPICE, the comparison should be interesting. Parameters for the delay calculation are typical value for 0.8 μ m process technology.

Table 5.1 shows the number of FFs connected to a clock distribution logic and calculation time with our initial and detail clock routers. The calculation time of our initial clock router is almost proportional to the number of clock nets and the area in which buffers and flipflops are distributed. Our detail clock router required extra time to search the locations where routes of ordinary signals didn't exist and U-shaped modifications were possible, but the total time was not so long.

Table 5.1 Calculation Time

	No. of FFs	Clock Tree Level	No. of Raw Gates	Calc. Time (s)	
				Initial Clock	Detail Clock
Case 1	1793	3	63k	154	282
Case 2	50	2	154k	150	259
Case 3	132	2	154k	188	459
Case 4	3747	4	282k	1385	19861

Table 5.2 shows the maximum path delay and clock skew in a clock distribution logic comparing our initial clock router with SPICE when we didn't execute our detail clock router. In other words, it shows the result of the first case in the third step. Because our initial clock router doesn't consider PC-capacitance of other segments, the maximum path delay of our initial clock router was 0.46 - 0.72 when the value of SPICE was equal to 1. As for clock skew, it surprised us that the skew with SPICE was 20 - 100 times larger. From this result, it isn't too much to say that conventional clock routers that don't take PC-capacitance into consideration cannot be practical for actual high speed LSI design. This surprising difference will become even greater in deeper submicron technology.

Table 5.3 shows the same comparison as above between our detail clock router and SPICE. In this case, it shows the result of the second case in the third step. The maximum path delay of our detail clock router was very close to the SPICE

Table 5.2 Timing Difference (ps)
Without PC-capacitance

	Delay in Initial Clock Router		Delay in SPICE	
	Max Delay	Skew	Max Delay	Skew
Case 1	2375	20	3597	387
Case 2	580	9	960	220
Case 3	1101	6	2379	624
Case 4	4293	27	5985	646

Table 5.3 Timing Difference (ps)
With PC-capacitance

	Delay in Detail Clock Router		Delay in SPICE	
	Max Delay	Skew	Max Delay	Skew
Case 1	3412	29	3805	109
Case 2	870	13	964	19
Case 3	2289	32	2364	104
Case 4	6084	69	6559	238

value and within the range 0.90 - 0.97 when the value of SPICE was equal to 1. And the difference in the clock skew was reduced to 1.5 - 3.7 times. We think this difference reflects the difference in the Rubinstein[11] and SPICE delay precisions because the extracted capacitance in both programs was almost the same. This results confirmed that PC-capacitance had to be taken into account in clock routing to make it practical for the design of actual high performance LSIs. Finally, the value of clock skew by SPICE was almost within 240ps. Therefore, our clock router is practical for the design of more than 200 MHz LSIs.

6. Conclusion

We have proposed a new clock routing method that accounts for the parallel and cross capacitance of other segments. Our parallel and cross capacitance extraction model is similar to that of typical load extraction tools. From our experimetal results, we conclude that (1) there was a big difference in the value of the clock path delay and clock skew between our clock router and SPICE after layout if we didn't consider parallel and cross capacitance with other segments, (2) our detail clock router, with consideration of the parallel and cross capacitance of other segments and U-shaped route modification, could reduce the clock skew in SPICE, and the resultant clock skew confirms the practicality of using our clock router to design LSIs of more than 200 MHz.

We believe that the proposed capacitance extraction model is valid if the number of routing layers is increased more than three because the parallel capacitance is only to be considered among segments in the same layer, the cross capacitance is considered between every pair of different layer segments and the capacitance shielding against the substrate by the lower layer segment is modeled.

In our future work, we plan to apply our new clock router to the multi-phase clock and to minimize the clock path delay by reducing the clock net length.

7. References

[1]R.S. Tsay, "Exact Zero Skew", ICCAD'91, pp336-339, 1991

[2]T.H. Chao, Y.C. Hsu & J.M. Ho, "Zero SkewClock Net Routing", 29th DA Conf., pp518-523,1992

[3]S. Pullela, N. Menezes & L.T. Pillage, "Reliable Non-Zero Skew Clock Trees Using Wire Width Optimization", 30th DA Conf., pp165-170, 1993

[4]S. Pullela, N. Menezes, J. Owar & L.T. Pillage,"Skew and Delay Optimization for Reliable Buffered Clock Trees", ICCAD'93, pp556-562, 1993

[5]M. Edahiro,"Delay Minimization for Zero-Skew Routing", ICCAD'93, pp563-566, 1993

[6]J. Cheng & C.K. Cheng, "Skew Sensitivity Minimization of Buffered Clock Tree", ICCAD'94, pp280-283, 1994

[7]M. Seki, et.al., "A Specified Delay Accomplishing Clock Router Using Multiple Layers", ICCAD'94, pp289-292, 1994

[8]K. Chaudhary, A. Onozawa & E.S. Kuh, "A Spacing Algorithm for Performance Enhancement and Cross-Talk", ICCAD'93, pp697-702, 1993

[9]T. Gao & C.L. Liu, "Minimum Crosstalk Channel Routing", ICCAD'94, pp610-615, 1994

[10]D.A. Kirkpatrick & A.L. Sangiovanni-Vincentelli, "Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems", ICCAD'94, pp616-620, 1994

[11]J. Rubinstein, P. Penfield & M.A. Horowitz, "Signal Delay in RC Tree Networks", IEEE Trans. on CAD, vol. CAD-2, No.3, pp202-211, July, 1983

[12]W.C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers", J. of Applied Physics, vol.19, No.1, pp55-63, January, 1948

[13]H. Kitazawa,"A Line Search Method for High Routing Rate", J. of Information Processing Society of Japan, vol.26, No.11, pp1366-1375, 1985 (in Japanese)