Abstract: Inter-wire coupling continues to become a more significant portion of the total wire-capacitance in deep submicron designs. Coupling introduces noise that has 2 most important aspects: functional noise and delay noise. Additional delay causes timing closure problems, and noise destroys signal integrity. Fixing functional and delay-noise violations are time consuming, however, due to the iterative routing and analysis need with current crosstalk-avoidance design flows. The ultimate goal for crosstalk avoidance routing is to reduce delay uncertainty as well as coupling noise. In this paper, we present a method called "net classing" to avoid both types of noise during the global routing phase. The basic idea is to have a global view of interconnect behavior. We created a path analyzer to identify timing critical nets and a coupling analyzer to identify noise critical nets. We feed the net class information into Cadence tool to complete detail routing and noise analysis. Comparing to a crosstalk avoidance routing flow, CeltIc analysis of detail-routed results showed that net classes help reduce timing uncertainty by 6.47% and maximum noise peak by 4.5%Vdd on average with 0.25um technology node, 3.76% on timing uncertainty and 2.95%Vdd on peak noise in 0.13um technology node. We also compare the novel 3D design with conventional 2D design and find an average 3.1% reduction on timing uncertainty and 2.13%Vdd reduction on peak noise with a 20% reduction in worst case delay.

I. INTRODUCTION

Capacitance coupling between neighboring nets is dominating other capacitance in deep sub-micron (DSM) technology. We must consider coupling for its impact on functional correctness and timing convergence. Coupling began to have influence on timing and noise with 0.25 µm technologies [7], and the effect becomes even worse in later technology because of increase in metal aspect ratio. Delay-noise [17] causes timing closure problems in high performance IC design and functional-noise [17] can change the state of a flip-flop and waste power.

A standard commercial design-flow has emerged to address these problems. Crosstalk-aware routers attempt to avoid delay-noise by spacing timing-critical nets from other nets to reduce uncertainty. Functional-noise avoidance uses a similar strategy. Therefore, both timing- and noise-critical nets require extra routing resources. Some novel routers will also swap a wire's metal layer, but this is less common, since layer resources are limited and higher metal layers are usually reserved for global nets or power nets. Shielding can also be used to reduce crosstalk-noise, but spacing is favored, since it uses similar routing resources but does not increase a net's ground capacitance. However, empty tracks are not always available especially in congested areas.

A graph of the industry-standard crosstalk-aware routing flow is shown in Figure 1 (modified slightly from flow presented in [18]). After a design has been placed, early noise-prevention techniques are applied such as driver sizing to reduce slew-rates on nets that are likely to be strongly coupled. Only the vaguest idea of coupling is available at this stage of the flow, because no wire-adjacency information is known. Next, the design is routed, parasitics are extracted, and functional noise violations are found and corrected by a

![Figure 1. Standard Crosstalk-Aware Routing Flow.](image-url)
succession of routing passes. Afterwards, the switching windows (the times that adjacent wires are likely to be switching), are determined by static timing analysis. Based on these windows, incremental delay-uncertainty is determined for each net. Nets on paths that exceed their timing constraints are then spaced and re-routed in an attempt to meet the constraints.

One difficulty of the traditional crosstalk fixing flow is that functional and delay noises are handled separately. Since each violation typically requires extra spacing, care must be taken to ensure that a delay-noise fix does not affect a functional-noise fix and vice versa. Another difficulty is that it requires a great deal of designer effort to figure out how to set the various options in the flow such that performance is optimized. A designer must generally experiment with the options on a number of attempts to get the right set for their particular design.

To divide nets into different classes for a detail crosstalk control is an option in cadence router. However, setup such net classes may require a lot of expertise and design iterations. Our goal in this work is to reduce designer effort and the effects of crosstalk by analyzing the design at an early noise-prevention phase and developing a net-classing strategy that is optimal in a global sense. We developed a global router to estimate wire topology and an analyzer to shuffle wires between the classes in search of a global optimum. The output is a DEF file with class assignments that can be used by Cadence Warp Router.

The paper is organized as follows. Section 2 presents the model to estimate timing uncertainty and noise on coupled pairs. Section 3 introduces the net classification scheme. Section 4 presents the method to find critical nets. Section 5 shows experimental results of our model and crosstalk in different routing schemes, it also briefly show how crosstalk is relieved in 3D ICs.

II. TIMING AND NOISE MODELS

The most common approach to estimate delay-noise is to use a “switching factor”, which is multiplied with the coupling capacitance to convert it to an equivalent ground capacitance. A switching factor of 2 is widely accepted as an industry standard, however there is much evidence to suggest that this is not a sufficient model. Kahng et al. [11] showed that coupling becomes more severe as slew rate increases, and that a switching factor as high as 3 would need to be used.

There has been a great deal of work on methods to improve crosstalk estimation beyond the switching factor approximation. Closed-form formulas based on π2 and π4 RC models to analyze peak noise were presented in [1] and [2]. These models provide an accurate way to analyze peak noise between two coupling nets. These methods do allow calculation of crosstalk effects in constant time, but the calculations have no effects on design quality without incorporating them into a design flow. Kahng, et al. [10] presented another approach to estimate crosstalk-noise, but this method is too time consuming to be incorporated in pre-route estimation.

Pre-route noise and timing uncertainty estimation must be fast yet with satisfactory accuracy. Here we show the calculation of functional noise voltage. Consider two coupling nets shown in figure (2a). The equivalent circuit can be represented as figure (2b). We name the elements in this circuit using the scheme presented in [1], where \( R_h \) represents the holding resistance (the output resistance of each driver). Each aggressor is assumed to couple to each victim over a coupling-length \( L_c \), and the remaining lengths of the aggressor and victim are labeled \( L_{ax}, L_{vx}, L_{ay}, \) and \( L_{vy} \). Assume parasitic resistance and capacitance of unit length is \( r \) and \( c \), we have

\[
R_{ax} = r(L_{ax} + L_c/2) \\
R_{ay} = r(L_{ay} + L_c/2) \\
C_{ax} = c(L_{ax} + L_c/2)/2 + C_{oa} \\
C_{ay} = c(L_{ay} + L_c/2)/2 + C_{oa} \\
C_{ai} = c(L_{ai} + L_{ax} + L_c)/2 \\
C_{ai} = c(L_{ai} + L_{ay} + L_c)/2 \\
\]

where \( C_{oa} \) and \( C_{La} \) are the output capacitance of the aggressor driver and input-capacitance for the loading gate. Similar values are found for the victim net. We can solve a linear equation series (2) to get relationship between outputs and inputs in frequency domain. To model noise, we assume victim \( V_4 \) stays low.

\[
V_4 = V_a + R_{ai}(V_S C_{ai} + V_a C_{i2} + (V_b - V_a) S C + V_{out} C_{i3}) \\
V_4 = V_a + R_{ai}(V_S C_{i2} + (V_b - V_a) S C + V_{out} C_{i3}) \\
V_b = V_{out} + R_{ai} V_S C_{i3} \\
0 = V_d + R_{xi}(V_S C_{xi} + V_{xi} C_{i2} + (V_e - V_{xi}) S C + V_{out} C_{i3}) \\
V_e = V_{out} + R_{xi}(V_S C_{i2} + (V_e - V_{xi}) S C + V_{out} C_{i3}) \\
V_e = V_{out} + R_{xi} V_S C_{i3} \\
\]

Solve (2) and re-labeling the coefficients, we got:

\[
V_{out} = \frac{a_1 s + a_2 s^2 + a_3 s^3}{b_0 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4 + b_5 s^5 + b_6 s^6} V_i(s) \\
H(s) = \frac{a_1 s + a_2 s^2 + a_3 s^3}{b_0 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4 + b_5 s^5 + b_6 s^6} \\
\]

Using dominant pole approximation as in [1] and [2].
When net1’s driver is high or low, we cannot solve for frequency poles. Although this technique can lead to a moment matching technique is used to analyze a few low frequency poles. Although this technique can lead to an analytical expression for delay, we cannot solve for the delay value explicitly. Iterative calculation must be used, and the computation penalty is too high. Instead, we used an equivalent ground capacitance model for delay estimation which can be determined quickly. Elmore delay is used to roughly estimate the timing uncertainty since we have got equivalent ground capacitance with satisfactory accuracy.

Equivalent ground capacitance is related to the slew rate (20%-80% transition time) of the coupling nodes. As shown in figure 3, we assume \( t_{r1} < t_{r2} \) and unit ramp input waveforms. At point A, the voltage difference on \( C_c \) is \(-V_A\), while at point B, the voltage difference on \( C_c \) is \( 1-V_B \).

Thus, a voltage change during net1’s transition is \( \Delta V = V_A - V_B \). Since \( V_A - V_B = t_{r1}/t_{r2} \), we can say that as net1 starts to switch, the total charge change is

\[
\Delta Q_{total} = C_c \times \Delta V = (1 + V_A - V_B) C_c = (1 + t_{r1}/t_{r2}) C_c
\]

\( t_{r1} \) and \( t_{r2} \) are rise/fall time of net1 and net2 at the coupling nodes. The total charge change should be \( \Delta Q_{total} = 2 C_c \). When net1’s driver is high or low, it has no effect on \( C_c \) any more, so the remaining charge \( \Delta Q_1 = (1 - t_{r1}/t_{r2}) C_c \) is provided by net2’s driver.

Assume we know slew rate of node B, E in figure 2b, voltage on coupling point is:

\[
V_{coupling \_ points} = \begin{cases} 
\frac{1}{t_r} \left( t_d (1 - e^{-t/d}) + t \right), & 0 \leq t \leq t_r \\
\frac{1}{t_r} \left( t_d (e^{-t/d} - e^{-t_{r1}/d}) + t_r \right), & t > t_r 
\end{cases}
\]

If \( V_{coupling \_ points}(t) < 0.2 \), \( \text{slew} = t_d \ln 4 = 1.39 t_d \).
If \( V_{coupling \_ points}(t) > 0.8 \), \( \text{slew} = 0.44 t_d - 0.122 t_d \).
In other cases,

\[
\text{slew} = t_d \ln \left( \frac{t_d (e^{t_d/d} - 1)}{0.2 t_d} \right) - 0.2 t_r - 0.18 t_d \]
within 2 iterations. Non-monotonic waveform would occur when slew rates of node B, E in figure (2b) are highly unbalanced. This introduces error to both aggressor and victim. The equivalent ground capacitance is given in equation (11)

$$\text{equivalent } _{\text{cap}_{\text{net1}}} = (1 + \frac{L_1}{t_{r2}})C_c = (1 + \frac{slew1}{slew2})C_c$$

$$\text{equivalent } _{\text{cap}_{\text{net2}}} = (3 - \frac{L_1}{t_{r2}})C_c = (3 - \frac{slew1}{slew2})C_c$$

III. WIRE PLANNING

The basic difficulty of analyzing delay-noise at the global-route phase is that wire adjacencies are not known. Therefore, much effort in this area has focused on relieving overall congestion [4, 6]. Kao and Parng [4] presented an algorithm to make cross point assignments (CPA) in G-cells to provide a better idea of the congestion than is available from a detail route. Hadsell and Madden [6] demonstrated a rip-and-replace route approach with congestion estimation. These approaches can reduce the effects of crosstalk by creating more space between wires on average, but they devote routing resources equally to all nets. Zhou and Wong [8] introduced routing criteria based on a simple functional-noise estimation model. Their work, however, uses a very simple estimator and cannot handle variation in slew rates or delay-noise. Our approach is similar, but has been enhanced to handle these effects.

We propose to reduce both functional noise and delay-noise at the same time in higher level abstraction. We created a heuristic critical-net analyzer that divides nets into three different classes: timing-critical nets (class 1), noise-critical nets (class 2), and normal nets (class 3). Timing-critical nets are basically long nets in critical paths. Noise critical nets are aggressors that introduce large functional noise to its victims or large delay uncertainty to class1 nets. Normal nets are those who have no crosstalk effect on their victims.

The critical-net analyzer is illustrated in figure3. As shown, timing-critical nets are assigned simply by analyzing path-delays. The global router then assigns nets to GCells, and the coupling-analyzer estimates coupling effects for each pair of nets by assuming the minimum spacing and counting the number of common Gcell edges they share. Normal nets are all the other nets that are not considered timing or noise critical. These nets don't need extra care and can be routed in traditional way.

![Figure3. Our Net-Classification Design Flow](image)

Based on these net classifications, we then set routing rules for commercial global- and detail-routers. Currently, we use the following routing rules:

1. **Class1 nets repel nets in classes 1, 2, and 3.** This rule gives as much spacing as possible for timing-critical nets to minimize delay uncertainty as well as static delay.
2. **Class2 nets repel nets in classes 1 and 2 only.** Since class2 nets can potentially cause timing closure or signal integrity problems, they should be separated from class1 or class2 nets. Class3 nets, however, can be adjacent, because they can act as shield.

The results in section V show that the use of these routing rules with Warp Route effectively reduced the functional- and delay-noise below that which was achieved with a crosstalk-avoidance flow.

IV. EXPERIMENT IMPLEMENTATION

The most important thing for net classing is to find out which nets belong to which class. To find out Class1 nets, we need to have path delay information. The timing estimation could be either from synthesis result or Encounter trial route after placement. Figure5 (a) gives the diagram of finding Class1 nets from synthesis report and figure5 (b) presents how to find Class1 nets with trail route timing estimation. In this experiment, we use the process described in (5a).

We have a simple global router to analyze routing candidate of each net to find out Class2 nets. This analysis assumes largest coupling possibility. We count the common Gcell edges for any two nets. The
is calculated based on placement information. Path delay from either prototype routing or design compiler is read in and an estimated worst case coupling delay is added to the initial static path delay. A net in critical path would be a class 1 nets if it exceeds threshold length. The length is determined by the uncertainty we allow.

The second step does a global route. The global router is implemented using Hadlock’s Algorithm [16] with a Gcell concept. Our global router generates an estimation of wire length and routing topology. Though no accurate information about neighboring nets is available, an RC network as in figure (2b) is estimated using the number of common Gcell-edge crossings for any possible pair of nets.

In the last step, we created a coupling analyzer to perform timing and noise analysis based on global routing information. Assuming minimum spacing, coupling capacitance is estimated based on the number of common Gcell-edge crossings. This analyzer will analyze both delay uncertainty and noise for any reference net. It also decides class 2 and class 3 nets. The pseudo code for coupling analyzer is shown in figure (4b). When a net shares more than 5 common Gcell edges with reference net, the analyzer will invoke

```plaintext
compute # of common Gcell edges that 2 nets may share
assign Class 1 nets
for each net in net list
    if common Gcell edges >= pre-assigned constant
        if reference net != Class 1 net
            analyze noise to reference net
        else
            // reference net is a class 2 net
            analyze both noise and timing uncertainty to reference net
    if either noise or uncertainty >= threshold
        if aggressor != Class 1 net
            aggressor = Class 2 net
    if aggressor = Class 2 net
        aggressor = Class 3 net
```

V. PSEUDO CODE

In this section, we present pseudo code used in our approach. The first step is to perform a static path analysis using path analyzer. The pseudo code of path analyzer is shown in figure (4a), a Manhattan distance

![Diagram](image-url)

Figure 6 identify Class 2 nets

After all nets are assigned a net class, a def file with placement, connection and net class information is dumped out. This def file is very similar to a placed def file created by Encounter except that each net ends up with a line “+ XTALK n”, where n can be 1, 2 or 3 representing class 1, class 2 or class 3 respectively. Warp Router takes this def file as an input, then select option Wroute advanced->Crosstalk Options, select “Detailed Control of Crosstalk Class Spacing” and type “1+1|2|3, 2+1|2” in the text area. This provides the necessary net adjacency control as we hoped.

```plaintext
for each path in critical path list
    assume worst case RC delay on this net
    sum path delay and net delay
    if final path delay > pre-determined path delay
        for each net from longest net to shortest net
            if each net >= length threshold
                each net = Class 1 net
```

In the last step, we created a coupling analyzer to perform timing and noise analysis based on global routing information. Assuming minimum spacing, coupling capacitance is estimated based on the number of common Gcell-edge crossings. This analyzer will analyze both delay uncertainty and noise for any reference net. It also decides class 2 and class 3 nets. The pseudo code for coupling analyzer is shown in figure (4b). When a net shares more than 5 common Gcell edges with reference net, the analyzer will invoke

```plaintext
compute # of common Gcell edges that 2 nets may share
assign Class 1 nets
for each net in net class
    if common Gcell edges >= pre-assigned constant
        if reference net != Class 1 net
            analyze noise to reference net
        else
            // reference net is a class 2 net
            analyze both noise and timing uncertainty to reference net
    if either noise or uncertainty >= threshold
        if aggressor != Class 1 net
            aggressor = Class 2 net
    if aggressor = Class 2 net
        aggressor = Class 3 net
```

noise model if the reference is not class 1 nets. Otherwise, both noise and delay will be analyzed.
VI. EXPERIMENTAL RESULTS

To validate our approach, we ran four sets of experiments. The first set compared the peak-noise model in equation (8) to spice simulation. The second set compared our equivalent ground-capacitance model in equation (11) to the traditional assumption that equivalent ground capacitance can be estimated as 2Cg or 3Cg. The third set compares functional- and delay-noise of detail-routed layout for 3 benchmarks using our net-classifications with Warp Route to a crosstalk-avoidance flow. The last set compares 2 benchmarks implemented in novel 3D IC structure and conventional 2D IC. It demonstrates the possibility of extending existing cadence tool to complete 3D IC design. These runs were performed on a Sun Ultra-Sparc3 900MHz CPU with 16G memory. Designs are implemented in the 0.25um, 0.13um technology from TSMC and 0.18um Low Power SOI 3D technology from MIT Lincoln Lab.

**Experiment 1: Peak Noise Model Vs Spice Simulation**

This set of experiments showed how accurate our noise model is. To estimate noise from (8), we randomly generated 10000 circuits as shown in figure (2b). Rh1 and Rh2 are in the range of 50 to 3000Ω. Las, Lae, Lvs, Lve are greater than 30um, Lc is greater than 150um. Output drain capacitance Cda (CdV) is in range of 5ff to 50ff and rise time is in the range of 10ps to 500ps. Table1 showed the result. The error is defined as:

$$\text{error} = \frac{\text{estimated error} - \text{actual error}}{\text{actual error}} \times 100\%$$

<table>
<thead>
<tr>
<th>Error range</th>
<th>Within +/- 40%</th>
<th>Within +/- 20%</th>
<th>Within +/- 10%</th>
<th>Within +/- 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction of nets</td>
<td>99.9%</td>
<td>90.4%</td>
<td>68.9%</td>
<td>44%</td>
</tr>
<tr>
<td>Average</td>
<td>1.44%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>σ</td>
<td>11.7%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table1 noise Comparison

Figure7 shows the histogram of peak noise error. The model slightly overestimates the noise, with most error within +/-40%. This estimation is accurate enough to catch large noise on coupled nets. The estimation error appears larger than that reported in [2], but that analysis modeled the victim wire only, while our analysis models both the aggressor and victim wires.

**Experiment 2: Equivalent Ground Capacitance Model**

To demonstrate our equivalent ground capacitance model, we compared SPICE simulation of an aggressor-victim pair to victim-only circuits, which assumed that the coupling capacitance was referenced to ground. Traditionally, when referencing the coupling capacitance to ground, it is scaled by a "switching-factor" of 2 or 3. Table 2 shows the error when using these switching factors compared to using the slew-rate-determined switching-factor from equation (11). The set of circuits was identical to the one described for Experiment 1. Error is defined as:

$$\text{error} = \frac{\text{equivalent victim delay} - \text{entire network delay}}{\text{entire network delay}} \times 100\%$$

<table>
<thead>
<tr>
<th>Equivalent Cgnd</th>
<th>Mean</th>
<th>Max</th>
<th>σ</th>
<th>error&lt;20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Cc</td>
<td>28.41%</td>
<td>175.84%</td>
<td>19.15%</td>
<td>34.35%</td>
</tr>
<tr>
<td>2Cc</td>
<td>2.64%</td>
<td>95.13%</td>
<td>11.96%</td>
<td>90.80%</td>
</tr>
<tr>
<td>By slew</td>
<td>-0.48%</td>
<td>29.33%</td>
<td>5.02%</td>
<td>99.82%</td>
</tr>
</tbody>
</table>

Table2 delay comparison

These results show that the equivalent ground-capacitance model is off by less than 20% in 99.8% of the cases.

**Experiment 3: Net Class Route Vs. Commercial Flow**

To quantify the improvement of the net-classification scheme, we used it in conjunction with Warp Route to create detailed mask-layout without routing violations. Delay-uncertainty and noise-peak voltages were estimated with Celtic from Cadence. We then compared these results with a commercial crosstalk-avoidance flow, as described in section I. It is difficult to find benchmark circuits large enough to adequately demonstrate coupling effects, so we chose the largest that we could find that were easily synthesizable. These benchmarks include B18 and B19 from the ITC'99 suite [14] and one locally-developed benchmark, a 64-bit dot product with a 4-stage pipeline. The delay uncertainty is defined as:

$$\text{uncertainty} = \frac{\text{crosstalk} - \text{induced} - \text{delay}}{\text{static} - \text{path} - \text{delay}}$$

(12)

Table3a and 4a shows the absolute delays and runtime. Static delay represents the steady state critical path delay without coupling. Worst-case delay means the critical-path delay when delay-uncertainty is included. Due to crosstalk, the critical paths for static and worst-case may be different. Runtime indicates how
long it takes each routing scheme to complete detail routing. For the crosstalk-avoidance flow, we enable all options except buffer-insertion, gate-sizing or ECO placement. This was done for fairness, because other options could also be used with our scheme. The commercial crosstalk-avoidance flow spends considerably longer time than Warp Route alone and the actual timing depends on design complexity. Our Net Class Flow was much slower than the Warp Router, but the additional time is worthwhile considering the improvement and the ease-of-use from the user's perspective.

We focus our examination of the results on delay uncertainty as a percentage of static delay, thereby achieving some technology independence. Table 3b and 4b showed the delay uncertainty and noise-peak voltage, which is given as a fraction of the supply voltage. The results show that the commercial crosstalk-avoidance flow improved the delay uncertainty by 2.49% and noise peak-voltage by 6.53%Vdd on average over the Warp-Route result in 0.25 technology node. The net-classification approach, however, improved the delay uncertainty by 6.47% and noise peak-voltage by 4.5%Vdd beyond the commercial crosstalk avoidance flow with same technology. In 0.13 μm technology node, the crosstalk avoidance flow improved the delay uncertainty by 7.38% and noise peak-voltage by 18.25%Vdd on average over the Warp-Route. The net class approach improved 3.76% and 2.95%Vdd respectively beyond crosstalk avoidance flow.

**Experiment 4: 3D IC Vs 2D IC.**

The crosstalk effect in 3D IC is also briefly discussed here. We developed a module based partition and floorplan method to deal with 3D ICs and then use cadence place and route tool to complete the rest of it. Final design can pass DRC and LVS check with NCSU 3DIC Cadence Design Kit (NCSU 3DIC CDK, available for download). Parasitic is merged and annotated to Celtic for crosstalk analysis. For a fair comparison, we did not apply net class routing scheme in 3D IC. Results show a 3.1% improvement on timing uncertainty and 2.13%Vdd reduction on peak noise over 2D IC with crosstalk avoidance flow. The worst case delay is also reduced by 20% comparing to its 2D counterpart. Pipeline64 is not implemented in 3D IC because it is a flat design. Perform partition and floorplan on a flat design is very time consuming and usually cannot give an optimal result with our current 3D IC design flow.

**VI. CONCLUSION**

We have demonstrated a net class step can further improve SoC Encounter’s routing performance by creating three net classes in the global routing stage. Experimental results show that combining these net classes with repelling-rules in standard routers can lead to less delay and better signal integrity in the final design. This approach sacrifices some run time but reduces the need for user analysis and iteration during the flow. This makes our approach especially attractive for high-level designers who want a good estimate of delay without having access to an experienced physical designer. Also, we observed that SoC Encounter has the ability to handle novel 3D IC design which reduces path delay as well as crosstalk. Ultimately, we believe that more early classification of nets and estimation of crosstalk-noise is necessary for fast convergence in the design flow and improved designer productivity.

**ACKNOWLEDGEMENTS**

We would like to thank Cadence Design Systems for providing Warp Route and Celtic, which were essential to complete our design flow.

<table>
<thead>
<tr>
<th>Design</th>
<th>Supply=2.5V</th>
<th>total nets</th>
<th>Chip area (mm²)</th>
<th>Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime (sec)</th>
<th>Crosstalk avoidance flow static delay</th>
<th>worst case delay</th>
<th>runtime</th>
<th>Net class routing static delay</th>
<th>worst case delay</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>B19</td>
<td>186241</td>
<td>12.96</td>
<td>28.844</td>
<td>30.68</td>
<td>421</td>
<td>28.062</td>
<td>29.681</td>
<td>N/A</td>
<td>27.016</td>
<td>27.969</td>
<td>10715</td>
<td></td>
</tr>
<tr>
<td>Pipeline64</td>
<td>88422</td>
<td>6.55</td>
<td>10.037</td>
<td>12.376</td>
<td>225</td>
<td>10.857</td>
<td>12.814</td>
<td>N/A</td>
<td>8.890</td>
<td>9.250</td>
<td>3664</td>
<td></td>
</tr>
</tbody>
</table>

Table 3a delay value & run time of different routing schemes (0.25um)

<table>
<thead>
<tr>
<th>Design</th>
<th>Supply=2.5V</th>
<th>total nets</th>
<th>Warp Router uncertainty noise</th>
<th>Crosstalk avoidance uncertainty noise</th>
<th>Net class routing uncertainty noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>B18</td>
<td>92685</td>
<td>8.13%</td>
<td>23.6%Vdd</td>
<td>6.57%</td>
<td>13.2%Vdd</td>
</tr>
<tr>
<td>B19</td>
<td>186241</td>
<td>6.37%</td>
<td>18.4%Vdd</td>
<td>5.77%</td>
<td>11.6%Vdd</td>
</tr>
<tr>
<td>Pipeline64</td>
<td>88422</td>
<td>23.3%</td>
<td>16.4%Vdd</td>
<td>18.0%</td>
<td>14%Vdd</td>
</tr>
</tbody>
</table>

Table 3b relative uncertainty & noise of different routing schemes (0.25um)
<table>
<thead>
<tr>
<th>Design Supply=1.2V</th>
<th>Total nets</th>
<th>Chip area (mm²)</th>
<th>Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime (sec)</th>
<th>Crosstalk avoidance flow static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>Runtime</th>
<th>Net class routing static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>B18</td>
<td>54744</td>
<td>1.3</td>
<td>23.628</td>
<td>26.72</td>
<td>923</td>
<td>20.644</td>
<td>22.68</td>
<td>N/A</td>
<td>20.617</td>
<td>22.402</td>
<td>1915</td>
</tr>
<tr>
<td>B19</td>
<td>127312</td>
<td>2.89</td>
<td>17.042</td>
<td>20.21</td>
<td>1254</td>
<td>15.515</td>
<td>17.21</td>
<td>N/A</td>
<td>14.943</td>
<td>16.192</td>
<td>3694</td>
</tr>
<tr>
<td>Pipeline64</td>
<td>56035</td>
<td>1.59</td>
<td>4.908</td>
<td>6.234</td>
<td>473</td>
<td>5.088</td>
<td>5.842</td>
<td>N/A</td>
<td>4.370</td>
<td>4.688</td>
<td>884</td>
</tr>
</tbody>
</table>

Table4a delay value & run time of different routing schemes (0.13 um)

<table>
<thead>
<tr>
<th>Design Supply=1.2V</th>
<th>total nets</th>
<th>Chip area (mm²)</th>
<th>Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime (sec)</th>
<th>Crosstalk avoidance flow static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>Runtime</th>
<th>Net class routing static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>B18</td>
<td>54744</td>
<td>11.3%</td>
<td>27%Vdd</td>
<td>9.86%</td>
<td>15.07%Vdd</td>
<td>8.66%</td>
<td>14.09%Vdd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B19</td>
<td>127312</td>
<td>18.9%</td>
<td>54.83%Vdd</td>
<td>10.9%</td>
<td>20.95%Vdd</td>
<td>8.36%</td>
<td>11.41%Vdd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipeline64</td>
<td>56035</td>
<td>27.5%</td>
<td>30.35%Vdd</td>
<td>14.8%</td>
<td>21.40%Vdd</td>
<td>7.27%</td>
<td>23.06%Vdd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table4b relative uncertainty & noise of different routing schemes (0.13um)

<table>
<thead>
<tr>
<th>Design Supply=1.8V</th>
<th>total nets</th>
<th>Chip area (mm²)</th>
<th>Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime (sec)</th>
<th>Crosstalk avoidance flow static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>Runtime</th>
<th>3D IC + Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>B18</td>
<td>48468</td>
<td>5.23/1.96</td>
<td>26.637</td>
<td>30.76</td>
<td>1181</td>
<td>26.878</td>
<td>30.24</td>
<td>N/A</td>
<td>21.693</td>
<td>24.04</td>
<td>672</td>
</tr>
<tr>
<td>B19</td>
<td>96210</td>
<td>10.57/4.29</td>
<td>19.267</td>
<td>23.2</td>
<td>1831</td>
<td>19.047</td>
<td>21.64</td>
<td>N/A</td>
<td>17.218</td>
<td>18.61</td>
<td>1289</td>
</tr>
</tbody>
</table>

Table5a delay value & run time of different routing schemes (0.18um 3D SOI)

<table>
<thead>
<tr>
<th>Design Supply=1.5V</th>
<th>total nets</th>
<th>Chip area (mm²)</th>
<th>Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime (sec)</th>
<th>Crosstalk avoidance flow static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>Runtime</th>
<th>3D IC + Warp Router static delay (ns)</th>
<th>worst case delay (ns)</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>B18</td>
<td>48468</td>
<td>15.5%</td>
<td>20.8%Vdd</td>
<td>12.5%</td>
<td>12.13%Vdd</td>
<td>10.9%</td>
<td>14.21%Vdd</td>
<td>8.1%</td>
<td>13.23%Vdd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B19</td>
<td>96210</td>
<td>20.39%</td>
<td>28.92%Vdd</td>
<td>12.7%</td>
<td>28.92%Vdd</td>
<td>12.7%</td>
<td>28.92%Vdd</td>
<td>8.1%</td>
<td>13.23%Vdd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table5b relative uncertainty & noise of different routing schemes (0.18um 3D SOI)

Reference:
[5]. "http://www.opencores.org", opencore website