Introduction

Interconnect capacitance becomes larger when the thickness/minimum width of metal line becomes larger and spacing becomes smaller as CMOS process scales into 0.18um and below. Glitch and induced delay related to interconnect capacitance are more obvious than the previous process technology. In other words, noise coupled from the adjacent nets may result in functional failure more easily. Glitch will generate ripple of the output state when it's small, and will cause the output state change when it exceeds the cell noise threshold. The signal timing on the victim net may be speed up or slow down by the crosstalk coupling noise, and the timing violations may be generated. These impacts of crosstalk noise should not be ignored, and they can be prevented and reduced. However, the previous implementation flow is not enough to solve the crosstalk noise impacts. In this guide, we will show the model of crosstalk noise, how to prevent it, how to analyze it, and how to fix it in an efficient way.

Noise model

Crosstalk noise is induced by the interconnect capacitance between metal lines [Rose 99]. Fig.1 shows a simplified analysis of concise attributes of crosstalk noise.

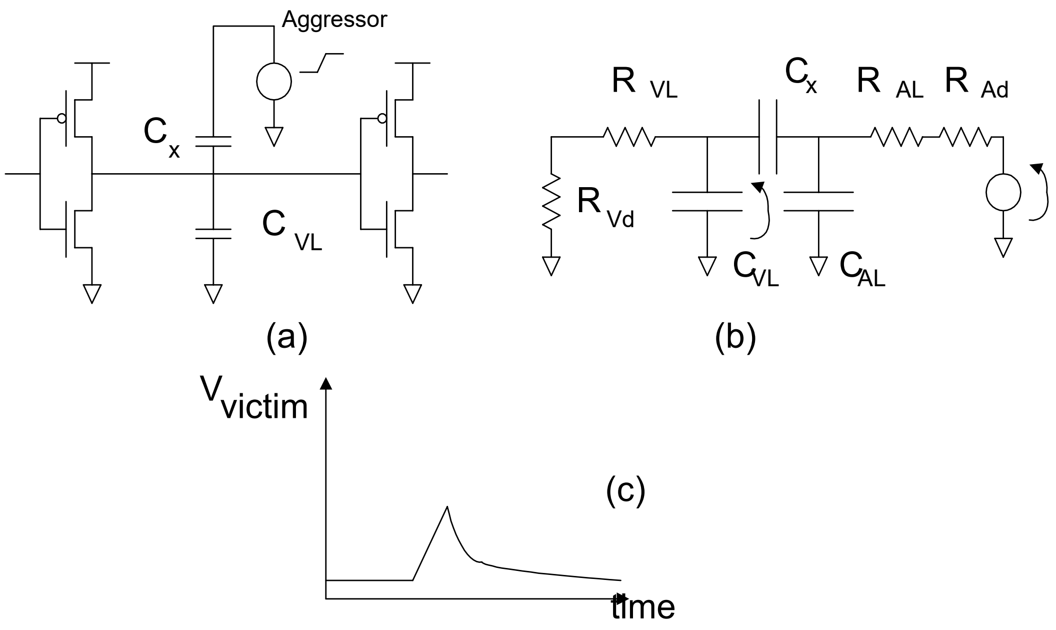

Figure 1. Crosstalk noise (a) coupling onto an evaluation node, (b) simplified equivalent circuit, and (c) pulse coupling noise waveform.

The victim net is attacked by an ideal aggressor with the transition time, t_r. The noise value can be approximated as the following equation:

$$v_{victim} = \frac{V_{dd}}{t_r}(R_{Vd} + R_{VL})C_x\left(1 - \exp\left(-\frac{t}{\tau_r}\right)\right) \qquad \text{for} \quad 0 \le t \le t_r$$

$$v_{victim} = \frac{V_{dd}}{t_r}(R_{Vd} + R_{VL})C_x\left(1 - \exp\left(-\frac{t_r}{\tau_r}\right)\right)\left(\exp\left(-\frac{t - t_r}{\tau_r}\right)\right) \qquad \text{for} \quad t > t_r$$

Where:

v_{victim}	: Victim noise value
V_{dd}	: Supply voltage
R_{Vd}	: Effective resistance of the victim driver
R_{VL}	: Effective resistance of the victim net

R_{Ad}	: Effective resistance of the aggressor driver
R_{AL}	: Effective resistance of the aggressor net
C_{VL}	: Lumped ground capacitance of the victim net
C_{AL}	: Lumped ground capacitance of the aggressor net
C_x	: Lumped coupling capacitance of the victim net
τ_r	: $(R_{Ad} + R_{AL})(C_{AL} + C_x) + (R_{Vd} + R_{VL})(C_{VL} + C_x)$

The peak value of crosstalk noise is proportional to the lumped coupling capacitance of the victim net, the effective driver resistance of the victim net, and the effective resistance of the victim net. The time constant, τ_r, is proportional to the induced delay, but the peak value of crosstalk noise may be larger when it is smaller. In other words, if the coupling capacitance is large, the glitch on the victim net is large; if the effective driver resistance is small, the peak noise value on the victim net is small. In the commercial routing tools, the basic concepts of the prevention and fixing of crosstalk noise are reducing the coupling capacitance and selecting the cell with reasonable driving strength. Eventually, these routing tools may reduce the impacts of crosstalk noise well. However, they may take long time to achieve the timing closure. Reducing the coupling capacitance and selecting the reasonable cell are the fundamental ways to solve the crosstalk issues, and are not the bottlenecks of the implementation flow. Based on the same strategies, the next sections will show the efficient methodology to apply these tools and achieve timing closure.

Noise Prevention

It will be too late to fix crosstalk noise in the post-routing stage. Such incremental changes may generate new violations of timing and congestion issues, and make the runtime of post-routing tool too long or unpredictable. Prevention in the front stages can reduce most potential crosstalk noise effectively.

- Synthesis stage

  The driving strength of a cell is an important factor of crosstalk noise prevention. According to the crosstalk noise modeling equation, the metal line driven by a weak cell may be attacked seriously when the adjacent nets switching, while the glitch of the net driven by a stronger cell is smaller when the adjacent net switching. On the other hand, the metal line driven by a too strong cell or with sharp transition curve may be a stronger attacker to the adjacent nets. Choosing the right cell with reasonable driving strength in the synthesis stage is the first step to accelerate the implementation flow. If the weak cell is used in the front stage, the routing tool will take time to optimize the timing path according to the crosstalk noise report and search/repair again. Set "don't used cell" and max transition time are used to select the suitable cells in the synthesis tool. Avoid using cells with weak driving strength can make sure most nets have a strong driver. However, it may increase signal delay, leakage power, and more attackers with sharp transition curves. Set constraint on max transition time is a better way to handle the driving strength of nets with different loading. In the design using 0.13um process technology, max transition time from 0.6ns to 1.2ns is suggested to set in the synthesis and optimization stage.

- Post-placement stage

  Because of the limitation of wire-load model, the synthesis tool may not choose the correct cell for the design. To prevent the potential crosstalk issues, the cell should be re-optimize with the physical location after placement. In First Encounter, "slew balance" and "congestion removal" are used to optimize the transition time; In Astro, use "Post-Placement optimization" to fix the max transition time; In Physical Compiler, set the max transition time constraint and fix them via "physopt". The RC calculation of each placement tool may be different from the real RC extraction tool, such as Star-RCXT. To get a robust result, the RC scaling factor in the placement tool, calculated after correlating with the real RC extraction tool, should be set a little conservative.

- Routing stage
  Crosstalk noise prevention option must be turned on in the global route, track assign, and detail route stages. It can reduce the overall runtime of fixing crosstalk noise obviously. Fixing crosstalk noise in search/repair stage only will take much longer runtime and the interconnect noise may not be reduced effectively. When the crosstalk noise prevention is turned on, routing tool can avoid assigning the long nets in the adjacent tracks according to the timing constraint. The earlier the crosstalk noise prevention option is turned on, the easier the routing tools achieve timing closure. If such option is turned on until in the post-route stage, routing tool will find the best routing patterns to meet the timing constraint without considering interconnect noise before the post-routing stage. In the post-routing stage, the routing tool may destroy the current routing patterns and perfect timing to reduce the interconnect noise, and then optimize the timing again. Eventually the routing tool may achieve the same timing violations, but the runtime may be much longer. The option of avoiding two nets routing in the parallel tracks more than a specified length is good to prevent crosstalk noise impacts in the false path set in timing constraint.

- Post-routing stage
  Crosstalk noise prevention option should be turned on when fixing the timing violation in the post-routing stage. The incremental changes after buffer sizing or buffer insertion may modify the routing patterns and timing of the design. Turn on the crosstalk noise prevention option will let the routing tool find the best routing path and reduce the interconnect noise at the same time. Turn on the crosstalk noise prevention option after the incremental changes and search/repair again may achieve the same timing closure eventually. However, the routing tool needs more runtime to fix the modified timing in the search/repair stage.

Noise Analysis

Dynamic simulation is not practical to check an entire digital IC (with millions of cell instances) for crosstalk effects. CeltIC employs a static noise analysis technique. Each cell instance in the design is analyzed from primary inputs to primary outputs, and a SPICE-like transistor-level simulator is used for noise induced delay analysis if necessary. The analysis of induced delay from crosstalk noise performed by CeltIC depends on the arrival times and slews rates of signals calculated by PrimeTime. Delay induced by crosstalk can only occur when the timing window for the victim and aggressor nets overlap. Re-characterized cell library for noise, logic constraints (relationships between signals) and timing constraints (timing windows during which signals can switch) are necessary for noise analysis in CeltIC. After calculation with the library, constraint and timing window, glitch report and delta SDF are generated to show the impact of crosstalk noise, and the ECO repair file is generated to fix these impacts.

Single operation condition of PrimeTime is used in our crosstalk analysis flow, best corner for hold time check and worst corner for setup time check.

● Noise library characterization

Before CeltIC can perform a comprehensive crosstalk analysis of a cell-based digital circuit, each cell in a cell library must be characterized by the makecdB utility or modeled using a UDN or ECHO to model the noise characteristics of the cell. The makecdB utility analyzes the transistor-level netlist of a cell for noise effects as seen from its input and output ports. It encapsulates the essential information inside a cell for noise analysis. Resistances, capacitances, and noise tolerance characteristics for each cell are taken into account. The characterized noise library typically has a .cdB extension, but this is not required.

The resultant cell library (.cdB file) generated by the makecdB utility contains a cell-level view and a transistor-level view. The cell-level view, or user-defined noise model (UDN), contains pin capacitance, calibrated input noise threshold, and nonlinear output drive strength. The transistor-level view contains the transistor cell subcircuit. The information in the .cdB file should be dumped in text format to make sure the setting and the further analysis is correct.

● Extracted RC in SPEF file

The SPEF file extracted by Star-RCXT with tsmc in-die process variation technology file is used in the analysis of crosstalk noise. To fix the crosstalk noise correctly, the resolution of extracting RC parameters should be set more accurate after large glitches or delay being reduced and check the report again with the more precise SPEF file.

● Timing window from PrimeTime

The delay uncertainty analysis performed by CeltIC is dependent on the arrival times and slew rates of signals get from PrimeTime. CeltIC uses the timing windows, arrival times, from PrimeTime to filter the unnecessary simultaneous switching between victim and aggressor nets, and report the crosstalk noise impacts according to the real signal switching scenarios.

● Timing group

Each timing window entry in the timing file refers to a reference clock. CeltIC uses timing windows to determine if signals can switch together to create a combined worst-case noise glitch. If signals have non-overlapping windows and belong to the same timing group, they are not combined. A timing group is defined by the set_timing_group command and is used to indicate clocks that are synchronous or asynchronous to each other. All clocks within a group are synchronous to each other and asynchronous to all other clocks outside the group. By default, all clocks belong to a single, synchronous group.

Ex: clk_125 is generated from clk_250, and it should be synchronous to
clk_250.
clk_125 and clk_250 should be showed in the timing window file.

```
set_timing_group {clk_250 clk_125}
```

Noise Calculation

There are two noise analysis modes in CeltIC, sensitivity mode and propagation mode. In sensitivity mode, if the noise peak calculated by the noise glitch calculation method is greater than a receiver's input noise threshold in analysis_noise stage then CeltIC will calculate the noise sensitivity of the receiver. Noise sensitivity is the basic rule that sensitivity mode deploys to determine if noise at the receiver is likely to cause a functional problem. It is based on the fact that if every logic gate in the circuit always filters noise then it is impossible for noise to create a functional failure. Noise waveform computation in CeltIC is performed with circuit simulation techniques considering fully distributed parasitic network for aggressors and victim. After noise waveform is computed, it is possible to simply check the noise peak voltage against a user defined global threshold in analysis_noise step and have all the victims with glitches exceeding this threshold reported as noise failures.

Noise Propagation

CeltIC performs transient simulations to propagate noise through multiple logic stages. Functional violations are reported when a gate output the glitch larger than a specified threshold or when the noise pulse propagates to flip-flops or latches. CeltIC considers the worst case combinable noise from multiple gate inputs, as well as the nonlinear combination of propagated noise with additional coupling in the downstream logic.

The accumulated noise waveform is then tested at the inputs of downstream flip-flops as shown in Figure 2.

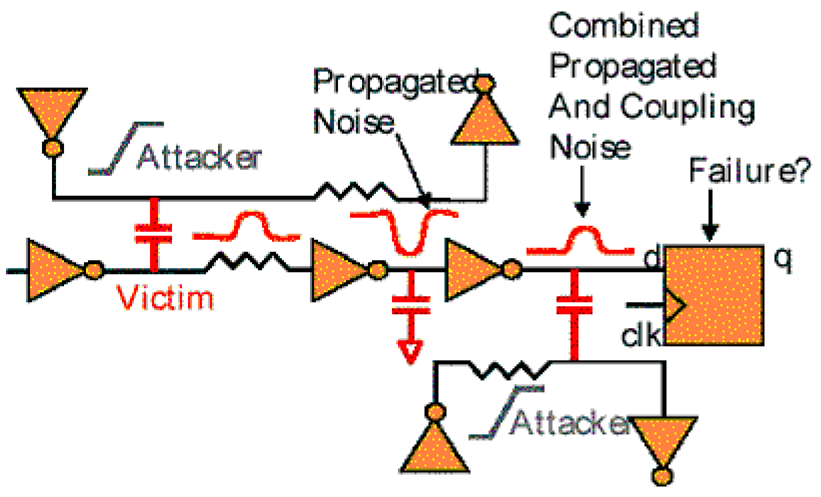

Figure 2. Noise Propagation

When a noise failure is reported at a latch (or flip-flop), CeltIC will report the combined noise due to both propagated and coupling noise. It will also identify the contribution from each noise source. The contribution of propagated noise is shown in CeltIC's output reports with the prp noise type.

● Sensitivity mode

To calculate the noise sensitivity of the receiver, CeltIC performs a transistor level AC sensitivity simulation of the receiver with the glitch noise calculated in the previous step. The worst case amplification of the receiver (dvout/dvin) is recorded as the sensitivity measure.

Sensitivity greater than 1 means that a gate is amplifying noise and 1 is the default value to evaluate the noise failure. Cell with sensitivity greater than 1 is reported as a noise failure with the default setting. This failure level can be changed to be more optimistic or pessimistic by setting the set_parm stability_thresh parameter in sensitivity mode.

Here is an example log of sensitivity mode with stability_thresh 1 :

```
**********************************************************************
analyze_noise:              SUMMARY
analyze_noise:   Number of cells in design = 364861
analyze_noise:   Number of nets analyzed = 459736
```

```
analyze_noise:   Number of stability failures = 3
analyze_noise:   Number of problem noisenets = 3
analyze_noise:   Number of ECO changes = 3
analyze_noise:   Number of delay ECO changes = 20
*************************************************************************
analyze_noise:   Analysis finished at Tue Apr 01 23:27:53 CST 2003


Sensitivity Histogram: (magnitude)
          NC:                 0
     [    0.0,      0.1 )    10480
     [    0.1,      0.2 )    95
     [    0.2,      0.3 )    48
     [    0.3,      0.4 )    24
     [    0.4,      0.5 )    7
     [    0.5,      0.6 )    4
     [    0.6,      0.7 )    3
     [    0.7,      0.8 )    4
     [    0.8,      0.9 )    1
     [    0.9,      1.0 )    0
     [    1.0,      1.5 ]    3

Peak Histogram: (percentage of vddnom @ 1100.0 mV)
     [    0.0,     10.0 )    0
     [   10.0,     20.0 )    0
     [   20.0,     30.0 )    9269
     [   30.0,     40.0 )    1320
     [   40.0,     50.0 ]    80
```

- ## Propagation mode

Alternatively, CeltIC can be used in the propagation mode, where failures are only reported at flip-flops or latches and that noise from all other cells is propagated. Propagated noise is calculated using the same transistor level simulation that is used to calculate noise sensitivity. Propagated noise will be later combined with coupling noise when analyzing the output of a victim's receivers. The sensitivity threshold in propagation mode means the receiver peak noise (noise/VDD) in the output of each cell. Threshold greater than 0.2 means the receiver peak noise is larger than 20% VDD, and 0.2 is the default value to evaluate the noise failure. In the path to flip-flop or latch, failures are reported as stability failures; in the path to output port, failures are reported including problem noise nets. The problem noise nets contain the failures in the path to flip-flop or latch and the path to output port. Received peak noise greater than 20% VDD is reported as a noise failure in the default mode. This failure level can be changed to be more optimistic or pessimistic by setting the set_parm stability_thresh parameter in propagation mode.

The default mode in CeltIC 4.1 is propagation mode.

Here is an example log of propagation mode with stability_thresh 0.2 :

```
*********************************************************************
analyze_noise:              SUMMARY
analyze_noise:   Number of cells in design = 384289
analyze_noise:   Number of nets analyzed = 417066
analyze_noise:   Number of stability failures = 11
analyze_noise:   Number of problem noisenets = 123
analyze_noise:   Number of ECO changes = 123
analyze_noise:   Number of delay ECO changes = 0
*********************************************************************
analyze_noise:   Analysis finished at Wed Apr 09 05:03:33 CST 2003

Receiver Peak Histogram: (percentage of vddnom @ 900.0 mV)
     [    0.0,     10.0 )    34339
     [   10.0,     20.0 )    16
     [   20.0,     30.0 )    4
     [   30.0,     40.0 )    3
     [   40.0,     50.0 )    0
     [   50.0,     60.0 )    0
     [   60.0,     70.0 )    0
     [   70.0,     80.0 )    0
     [   80.0,     90.0 )    1
```

```
    [   90.0,   100.0 )     3
    [ 100.0,   110.0 )     1

Peak Histogram: (percentage of vddnom @ 900.0 mV)
    [     0.0,     10.0 )     0
    [   10.0,     20.0 )     0
    [   20.0,     30.0 )     23909
    [   30.0,     40.0 )     8367
    [   40.0,     50.0 )     1878
    [   50.0,     60.0 )     199
    [   60.0,     70.0 )     11
    [   70.0,     80.0 )     2
    [   80.0,     90.0 )     0
    [   90.0,   100.0 ]     1
```

Noise Failure Determination

When a noise pulse propagates to the input of a latch or flip-flop, CeltIC needs to determine if the noise pulse will cause a catastrophic functional problem. CeltIC provides the user three possible choices, receiver input noise waveform, receiver output noise amplitude, or receiver sensitivity. Normally the noise at the receiver output is much smaller than the noise at the input, but if the input noise is big enough the output noise may be amplified to become even bigger. CeltIC performs an accurate transient simulation with the actual noise waveform on the receiver input and the proper loading on the receiver output. Noise sensitivity is a measure of the amplification of the noise by the receiver and is computed by monitoring the potential rate of change in the receiver output voltage to minor changes in the input noise waveform (dvout/dvin). As shown in Figure 3, CMOS logic gates such as the inverter are inherently noise immune. As long as a receiver is biased by noise into the sub-unity-gain regions of its transfer characteristic, noise will be attenuated and the gate will act to restore the correct logic level. If, however, noise biases the receiver into the high-gain region, then noise will be amplified and a functional failure could result.

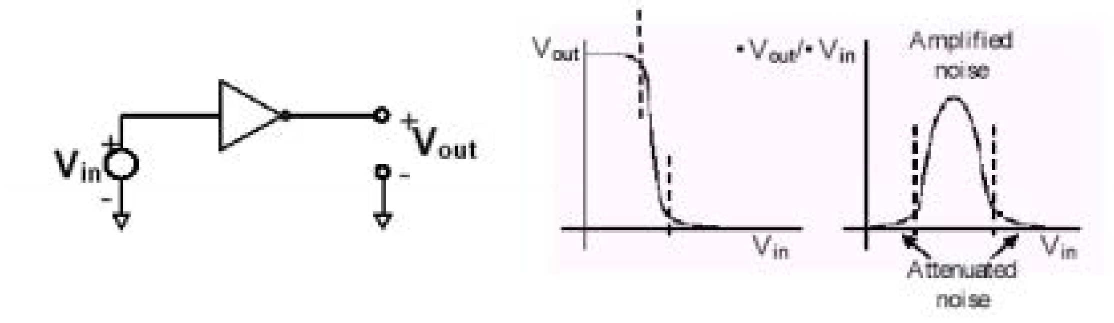

Figure 3. CMOS Inverter Noise Characteristics

To check sensitivity, a transistor level transient simulation of the receiver with the worst-case glitch waveform is performed. Sensitivity greater than unity gain (1.0) means that a receiver is amplifying noise and hence is deemed a noise failure under the noise sensitivity metric.

- ## Noise threshold

CeltIC analyzes all nets for noise starting from the input ports and working towards the output ports. Option "-noise_thresh" of command "analyze_noise" set the threshold of noise peak value. The noise larger than the specified threshold is used to calculated delta delay, and the other is suppressed. Default value of noise threshold is 20% of VDD. The number of analyzed nets will be increased when the noise threshold is reduced. The delta delay and glitch of major victim nets are almost the same. One small glitch will not hurt timing obviously. However, a lot of small attackers should be calculated.

- ## Virtual Attacker

A virtual attacker is an imaginary net introduced by CeltIC that is intended to represent the coupling effect of many small attackers on a victim net. Every victim net can potentially have a virtual attacker. The virtual attacker is named after the victim net. For example, for victim net XXX, the name of the virtual attacker is ~XXX. The use of a virtual attacker is a trade-off. If CeltIC treated every attacker explicitly and individually, the amount of computing and memory resources could be astronomical, because almost every net has many coupling capacitances connected to it, most of them very small. It is a mistake to completely ignore such small capacitances because when combined, they can cause trouble.

The most attackers for any victim net are usually very small and easily fall under the 5% threshold. These nets contributing 5% of VDD are called small attackers. Small attackers are too costly to be recorded and treated individually, so CeltIC creates a virtual attacker to represent the combined effects of these small attackers. Small attackers are combined to create a virtual attacker, and the combined virtual attacker has a coupling capacitance to the victim net as

$$C = 3\sqrt{C_1^2 + C_2^2 + ... + C_n^2}$$

The significance of this value is that given the n small attackers, each acting independentltly, if n is large, there is less than 4% probability that the attackers will attack with a combined effect larger than C.
To use the summation method, set the set_parm vmode control variable to 0 (the default). User can modify the combined capacitance by setting the set_parm vfactor control variable. For example, if user want to use 1.5 α to get less pessimism, set_parm vfactor to 0.5. (0.5 is 1/2, which, when multiplied with the 3 x α value, equals 1.5 x α).
The default mode ignores timing windows while assembling the virtual attacker and uses a default slew when driving the virtual attacker.
To compute a weighted average of the slews of the small coupling capacitance aggressor nets and use the computed slew to drive the virtual attacker, set the set_parm do_vatt_slew control variable to 1.

The virtual attacker is an engineering compromise that provides a time and memory savings, but sacrifices accuracy, reporting detail, and control on slew and timing window information. Because the virtual attacker is used only for small attackers, these sacrifices should be acceptable.

● Methodology

Receiver output noise larger than the stability threshold is reported in propagation mode, and dVout/dVin of each cell larger than the stability threshold is reported in sensitivity mode. User can understand receiver output noise more easily than dVout/dVin, and receiver output noise can be evaluated directly. Since the delta delay, glitch numbers and run time are almost the same in our test case with the setting mentioned in this application note, propagation mode is recommended in the noise analysis.