

**Interconnect Cross-Talk Induced Delay and Noise Glitch Analysis
for Embedded Microprocessor Designs**

**John Wei
Ashish Dixit
Steve Leibson**

**Tensilica, Inc.
(408) 566-1753
jwei@tensilica.com**

**INTERNATIONAL CADENCE USERS GROUP CONFERENCE
September 15-17, 2003
Manchester, NH**

Abstract

This paper details experiments that show that the CeltIC cross-talk analyzer can be integrated with existing place-and route (P&R), 3D-extraction, and static-timing-analysis (STA) tools. The CeltIC signal integrity (SI) analysis and fixing flow can generate SI ECO fixes for cross-talk noise glitches and can perform SI-aware timing signoff for cell-based, configurable, and extensible embedded microprocessor RTL designs.

The impact of coupling noise on maximum and minimum delays is analyzed in terms of setup- and hold-time violations, which complements STA. CeltIC coupled with STA ensures maximum accuracy and reflects interconnect cross-talk reality in the context of signal timing windows, slacks, and slews. The final, post-P&R STA is characterized by the incremental SDF generated by CeltIC, which is based on the SPEF extracted using Fire & Ice on the DEF generated by Silicon Ensemble.

CeltIC's fast run time and large capacity are demonstrated using three different embedded microprocessor configurations (simple processor, complex processor, and complex processor plus DSP extensions), using a state-of-the-art Artisan cell library and a characterized noise cell library (.cdB) for TSMC's 0.13um process technology. CeltIC's ECO capability to fix noise glitches is also demonstrated by creating an ECO DEF for the P&R tool to iteratively arrive at closure with respect to signal integrity.

1. Introduction

Timing signoff must consider design interconnect cross-talk effects by the 130nm technology node. The 2003 International Technology Roadmap for Semiconductors (ITRS) lists signal interference as one of the key design challenges starting at the 90nm technology node [1].

The existing hardware-implementation flow for Tensilica's configurable, extensible, and synthesizable Xtensa microprocessor core starts with RTL created by the Xtensa processor generator. The flow then goes through synthesis, P&R, 3D-extraction, and is followed by static timing analysis (STA). In addition to generating a configured processor, the Tensilica processor generator also provides estimates of design speed based on data pre-compiled from hundreds of post-layout designs done with the existing hardware implementation flow. Although cross talk effects have minimal impact on today's Xtensa designs, for the utmost accuracy with advanced process geometries (90nm and beyond), timing estimation will need to take interconnect cross-talk effects into account.

Cross talk's impact on performance varies and is unique to each processor generated because each Xtensa processor is unique, according to designers' application needs. Variation arises from configured processor function blocks and designer-defined processor extensions, which are all different. The comparison of STAs, with and without cross-talk effects, can guide designers as to whether or not to exert extra effort to reduce or fix cross-talk noise, based on severity.

This work demonstrates how CeltIC can be integrated with an existing hardware flow consisting of place and route, 3D-extraction, and STA hardware-implementation tools. CeltIC's analysis results, its fast run time, and large capacity are shown for three different embedded microprocessor configurations (simple processor, complex processor, and complex processor plus DSP extensions), using state-of-the-art Artisan cell and noise libraries for TSMC's 0.13um LV-FSG process technology. Because the demonstrated CeltIC signal-integrity analysis-and-fixing flow uses only commercial EDA tools, this successful integration can be readily replicated into other implementation flows that use other commercial EDA tools.

Finally, this work concludes by describing a few cross-talk minimizations and fixing techniques.

2. Embedded Microprocessor Design and Implementation Overview

(2.1) Embedded Microprocessor Designs

Designers can create unique processor configurations for their SOC applications using the Xtensa processor generator's pre-defined options. They can also add unique, designer-defined instruction-set and register extensions using the Tensilica Instruction Extension (TIE) language, as shown in Figure 1. The designer-specified configurations and designer-defined extensions cause the Xtensa processor generator to create a unique processor hardware design in RTL along with a suite of tailored software tools, models, and EDA support files and scripts.

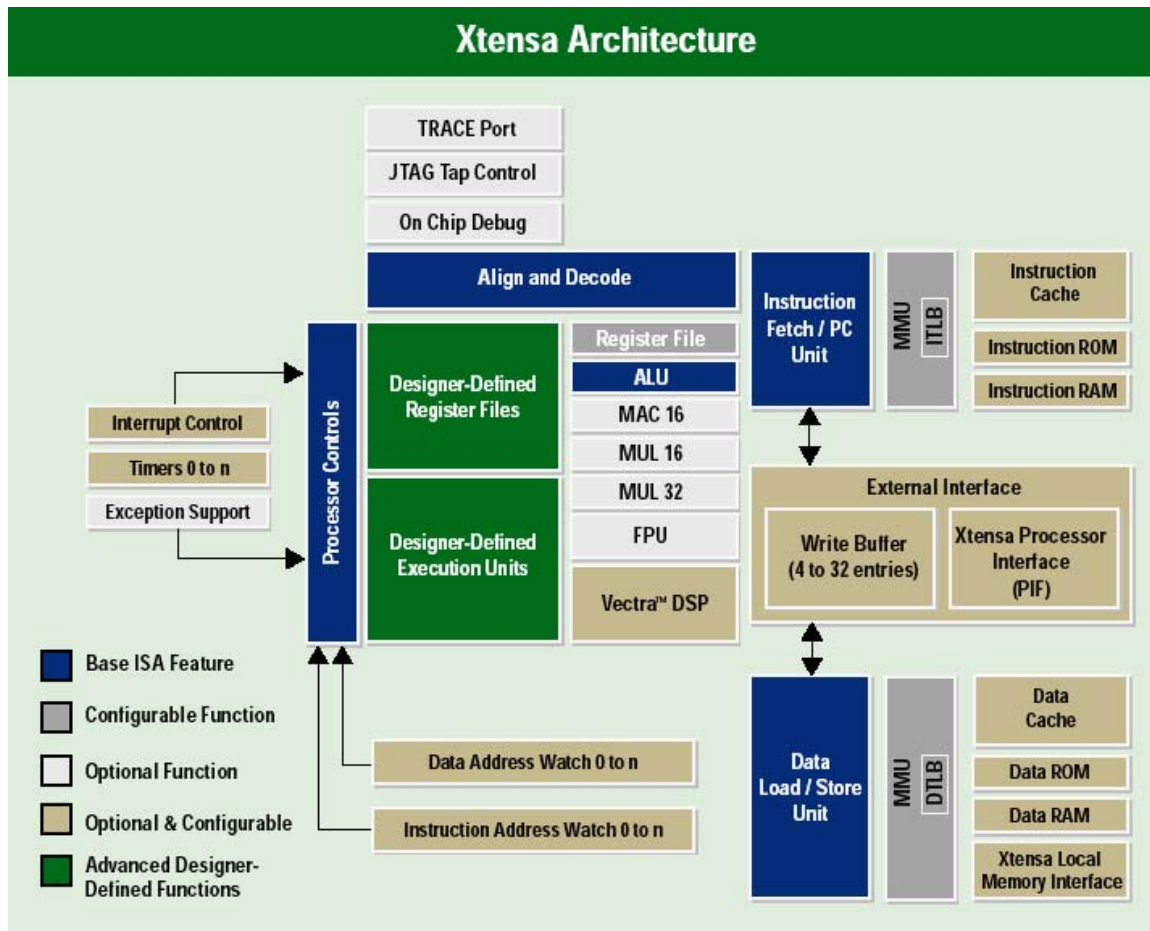


Figure 1: Soft IP – The Xtensa Configurable and Extensible Processor Architecture

(2.2) Hardware Implementation Flow

The hardware implementation flow for the Xtensa processor starts with synthesis, goes through place and route (P&R), 3D-extraction, and is followed by static timing analysis (STA), as shown on the left side of Figure 2. As a guide to hardware implementation, the Xtensa processor generator creates EDA scripts to accelerate the hardware-design flow and it also produces estimates of speed, area, and power based on data compiled from more than 200 post-layout designs of the processor architecture.

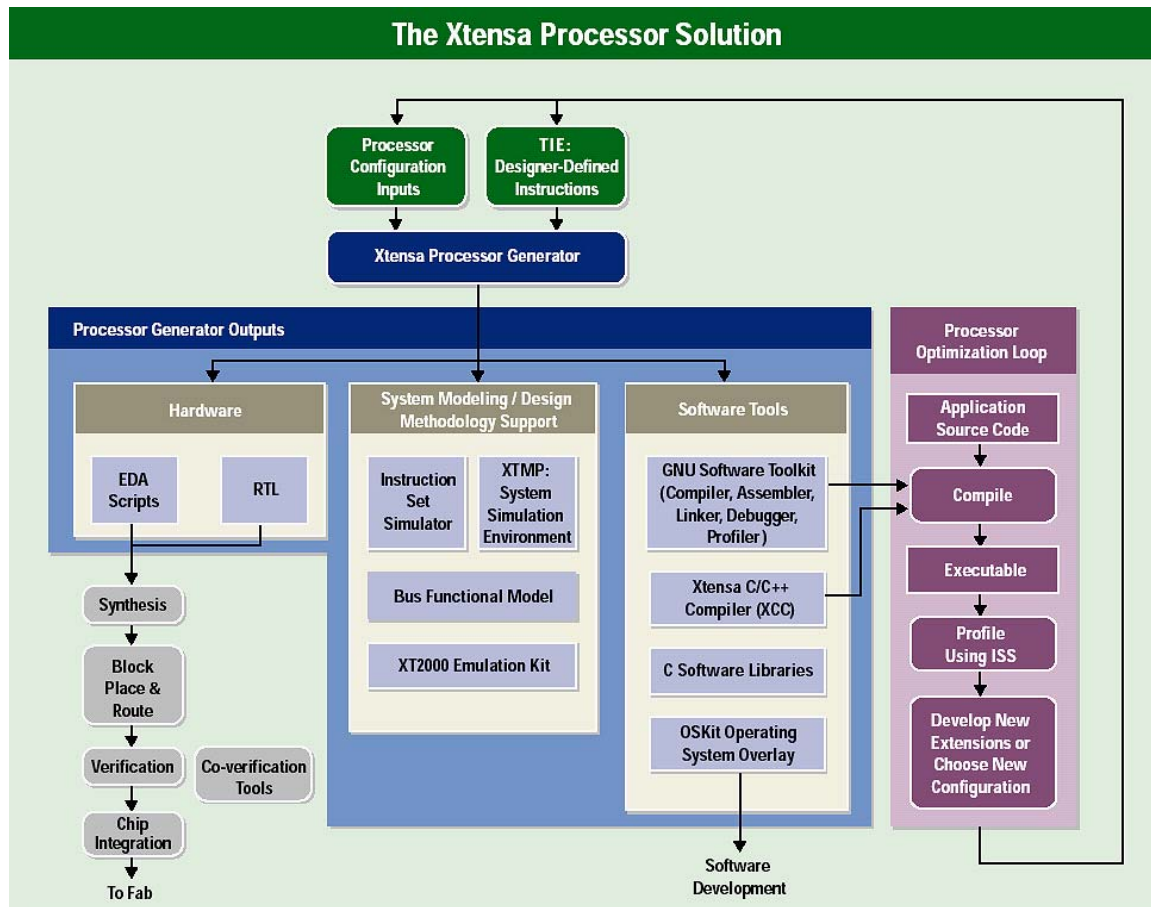


Figure 2: Xtensa Hardware and Software Implementation Flows

3 Interconnect Cross Talk Impact

As process geometries have shrunk through the progressing process-technology nodes, metal traces get narrower and taller to control wire resistance while holding the line on die-size increases. Although this change in aspect ratio controls wire resistance, it also increases the coupling capacitance between traces. Traces also grow closer together as geometries shrink, which further increases capacitance between wires. The increasing inter-trace capacitance, shown in Figure 3, increases the amount of cross talk between the traces. This increased cross talk decreases signal integrity, which in turn changes signal delays and causes glitches.

STA estimates enhanced by interconnect cross-talk effects will be important for future SOCs implemented in geometries of 90nm and below. The comparison of STA with and without cross-talk effects can guide designers as to whether or not to make the extra effort to reduce or eliminate cross-talk noise based on its expected severity and criticality.

Cross talk can be analyzed by computing the signal linkage between aggressor or attacker nets and victim nets. The attacker net carries a signal that couples to the victim net through the parasitic capacitance discussed above. To determine the effects that this cross talk will have on circuit operation, the resulting delays and logic levels for the victim nets must be computed.

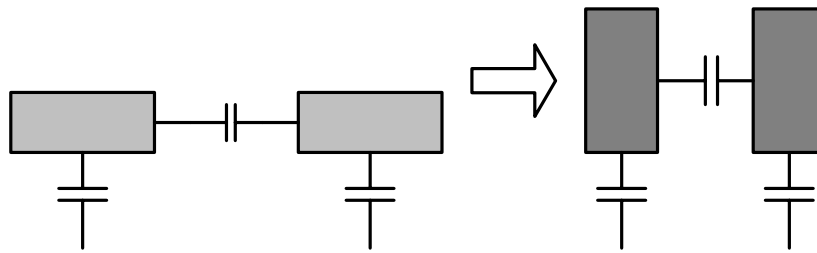


Figure 3: Progressive Process Technology Nodes – Increasing Wire Height/Width & Decreasing Spacing

(3.1) Design Timing Convergence – Delay Effect

Depending on the signal switching directions of the signals on the aggressor and victim nets, the impact of interconnect cross-talk can be either negative (increasing signal propagation time) or positive (decreasing signal-propagation time), which may cause setup- or hold-time violations. Figure 4 shows the case where cross-talk coupling noise impacts delay negatively, i.e. the cross talk delays the victim signal.

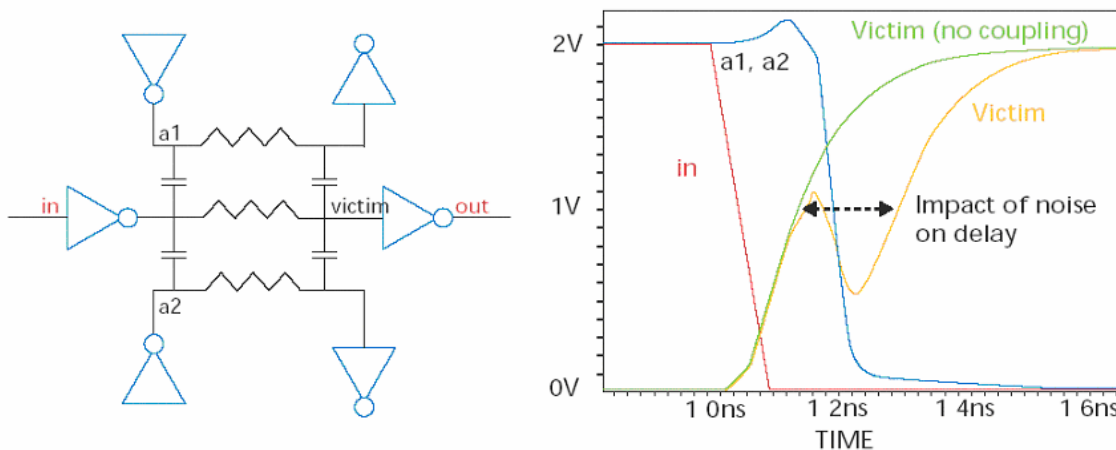


Figure 4: Coupling Noise Can Impact Delay – Negatively or Positively (Illustration Source: Cadence)

(3.2) Design Functional Failure – Noise Glitch Effect

In its extreme form, a noise glitch generated by cross-talk coupling can propagate and amplify while traveling along a path. This glitch can upset circuitry logic, as illustrated in Figure 5, which shows a glitch resetting a flip-flop, causing it to assume an incorrect state.

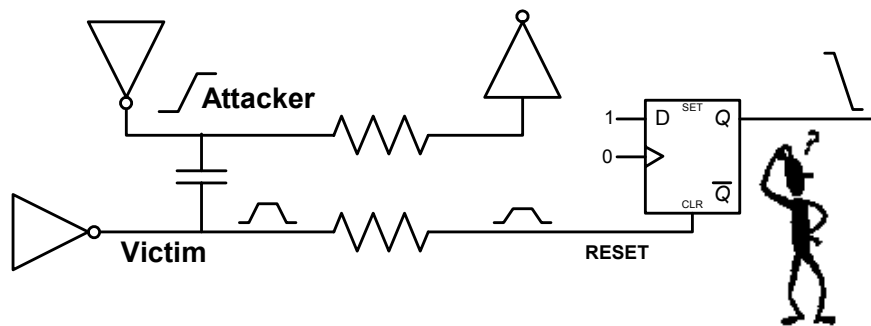


Figure 5: Coupling Noise Can Cause Functional Failures (Illustration Source: Cadence)

4 Using CeltIC with Tensilica's Hardware Implementation Flow

We have used CeltIC (Release CADMOS4.2) with Tensilica's hardware implementation flow experimentally, as shown in Figure 6. Taking the DEF placed and routed by Silicon Ensemble, Fire & Ice generates a SPEF with the design cell instances and the coupled_rc option on. Using this SPEF, CeltIC calculates the change in delay caused by coupling noise acting both with and against a transitioning signal, and outputs a delay uncertainty report in HTML format. With delay changes attributed to victim nets, CeltIC generates the incremental SDF for impacted nets.

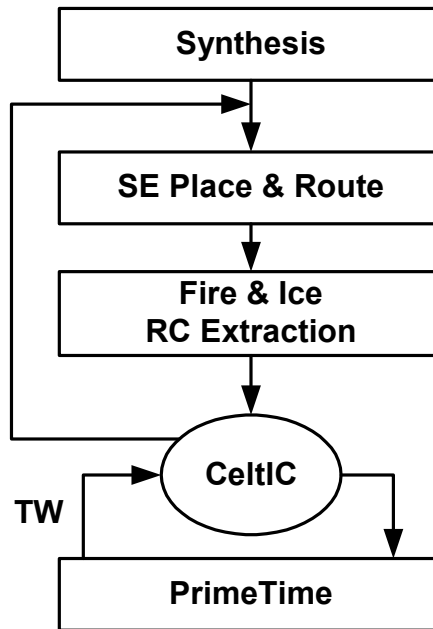


Figure 6: Using CeltIC with the Xtensa Hardware Implementation Flow

5 Experimental Implementation Details and CeltIC Results Analysis

Three microprocessors (simple processor, complex processor, and complex processor plus DSP extensions) were generated for this experiment using the Xtensa Processor Generator (Figure 1). The target clock periods for these designs are 2.5ns for the simple microprocessor and 2.86ns for both the complex microprocessor and the complex microprocessor with DSP extensions. The synthesis is optimized for $0.8 \times [\text{target clock period}]$ or, in other words, the over-constrain ratio is 0.8. SynthesisMaxTransition and ClockMaxTransition vary from 0.25ns, 0.50ns, 0.75ns, to 1.00ns. Clock skew is set at 0.18ns. Six layers of metal routing are used.

This work used TSMC's 130nm LV-FSG process, Artisan's standard cell library (tsmc13lvfsg_sc-x_2003q1v1), and its CeltIC noise library (tsmc13lv_sc-x_2003q1v1) characterized using LPE Spice netlist to ensure maximum accuracy. The tool versions used are: DC version 2002.05-2, SE version 5.4.122, Fire & Ice QX version 3.1.3a, PrimeTime version T-2002.09, CeltIC version CADMOS 4.2.

Fire & Ice QX extracts the SPEF with cell instance names from the P&R database (SE DEF routed with six layers of metal). The extraction_type was set to 'all coupled_rc' in the extraction command file to generate the coupled capacitance. The setvar spef_output was set to 'generic.'

The first CeltIC run using the SPEF generates an incremental SDF file, which is used in the first PrimeTime run to generate a timing-window (TW) file. A second CeltIC run reads in the timing-window file that contains the timing windows, slew, and slack information for all the switching nets. CeltIC uses the timing windows to determine the signals that can switch simultaneously, and those that cannot. CeltIC uses the slew information to switch attacking

signals. It performs different simulations for each time slot to create a worst-case noise scenario. A second PrimeTime run then uses the incremental SDF generated in this second CeltIC run to produce the final STA result, which usually converges. Each CeltIC run took between 15 minutes and one hour, running on a Sun E4500. These times were measured over the three test cases, from simple microprocessor to complex microprocessor with DSP extensions.

(5.1) CeltIC STA Results – Basic Microprocessor, Complex Microprocessor, and DSP Extension

Figures 7, 8, and 9 show the PrimeTime run results for the three configurable and extensible processor configurations. The figures also compare the PrimeTime results with and without cross-talk analysis. Max transition time is used as one of the cross-talk control variables to assess its impact on the final cross-talk effects.

Path Group	MaxTrans (ns)	PT - w/o CeltIC (ns)	PT - CeltIC w/o TW (ns)	PT - CeltIC + TW (ns)
Flop-to-Flop (max_delay)	0.25	-0.91	-0.91	-0.91
	0.50	-0.74	-0.86	-0.74
	0.75	-0.61	-0.86	-0.71
	1.00	-0.44	-0.69	-0.72
I/O (max_delay)	0.25	-0.67	-0.74	-0.68
	0.50	-0.60	-0.74	-0.74
	0.75	-0.48	-0.66	-0.67
	1.00	-0.43	-0.78	-0.82
Gated CLK (max_delay)	0.25	-0.33	-0.38	-0.33
	0.50	-0.54	-0.54	-0.54
	0.75	-0.53	-0.69	-0.61
	1.00	-0.55	-0.66	-0.55

Figure 7: Simple Microprocessor Timing Slack Violations (CLK period 2.5ns) – Gate Count 59,850
(Note: No min_delay or hold time violations for the three path groups; unity gate NAND2X1)

Path Group	MaxTrans (ns)	PT - w/o CeltIC (ns)	PT - CeltIC w/o TW (ns)	PT - CeltIC + TW (ns)
Flop-to-Flop (max_delay)	0.25	-2.27	-2.39	-2.27
	0.50	-2.17	-2.31	-2.22
	0.75	-1.94	-2.25	-2.14
I/O (max_delay)	0.25	-3.97	-4.11	-3.97
	0.50	-2.39	-2.40	-2.39
	0.75	-2.24	-2.40	-2.36
Gated CLK (max_delay)	0.25	-0.68	-0.68	-0.68
	0.50	-0.84	-0.84	-0.84
	0.75	-1.24	-1.24	-1.24
Gated CLK (min_delay)	0.25	-0.43	-0.39	-0.43
	0.50	0.00	0.00	0.00
	0.75	-0.03	-0.03	-0.03

Figure 8: Complex Microprocessor Timing Slack Violations (CLK period 2.86ns)
– Gate Count 139,811

Path Group	MaxTrans (ns)	PT - w/o Celtic (ns)	PT - Celtic w/o TW (ns)	PT - Celtic + TW (ns)
Flop-to-Flop (max_delay)	0.75	-1.32	-1.52	-1.49
I/O (max_delay)	0.75	-2.02	-2.19	-2.16
Gated CLK (max_delay)	0.75	-1.13	-1.33	-1.30
Flop-to-Flop (min_delay)	0.75	-0.06	-0.06	-0.06
Gated CLK (min_delay)	0.75	0.00	0.00	0.00

**Figure 9: Complex Microprocessor with DSP Timing Slack Violations (CLK period 2.86ns)
– Gate Count 293,794**

(5.2) Interconnect Cross-talk Effects Analysis – Setup Violations, Hold Violations, and Clock Gating

Interconnect cross-talk effects degrade estimated maximum clock speeds. For the simple microprocessor configuration (Figure 7), estimated design speed degraded from 322 to 312 MHz. For the complex microprocessor configuration (Figure 8), estimated design speed degraded from 196 to 192 MHz. For the complex-microprocessor-with-DSP configuration (Figure 9), estimated design speed degraded from 205 to 199 MHz. All three designs were constrained by a 0.75ns maximum transition time during synthesis and P&R. Examining the victim nets, the design speed was limited by either the internal flop-to-flop path (flop-to-flop group) or the processor I/O path (I/O group), depending on the processor configuration used.

Cross-talk effects create setup-time violations in each path group but have minimal impact on hold-time violations. The cross-talk effects on setup and hold times are expected to be significant when memory blocks are routed together with the processors, causing additional routing congestion. Cross-talk effects observed in this experiment with 130nm process parameters only amounted to a few percent, but they are expected to be far more significant when the microprocessor designs are implemented with 90nm process technologies.

To minimize operating power, the clock is gated at every possible place in Xtensa processors. Notably, clock gating also exhibits cross-talk-induced delay, particularly with the complex processor design with DSP extensions (Figure 9). As commonly implemented, the clock gating circuit is latch-based as shown in Figure 10. For this type of circuit, the gating enable signal (Gate or latch D-input) is sampled by a latch that closes on the rising edge of the clock. The latch subsequently guarantees the clock gating enable (Gate) hold time for the AND gate following the latch. The cross-talk delay impact on the clock-gating enable input (Gate) translates into the max delay violations shown in Figure 9, since the latch Q needs to meet the setup with respect to CLK' at the AND gate in Figure 10.

Because the latch is transparent when the clock is low, the gating enable input (Gate in Figure 10) needs to be stable only before the rising edge of CLK if the skews of CLK and CLK' are matched, as is the case when special clock-gating cells are used. When the skew match between CLK and CLK' is not guaranteed, as is the case when clock gating is built with discrete standard cells, the Gate enable signal needs to be stable for the latch-transparent, half-clock cycle of the gated clock. The cross-talk impact on the clock-gating enable input at either the rising edge of CLK or during the latch-transparent half cycle when CLK' skews earlier than CLK, can translate into glitches at the clock output (labeled Gated Clock in Figure 10). To ensure proper processor operation, it is essential to eliminate cross-talk-induced glitches on the clock-gating enable and on the clock tree itself.

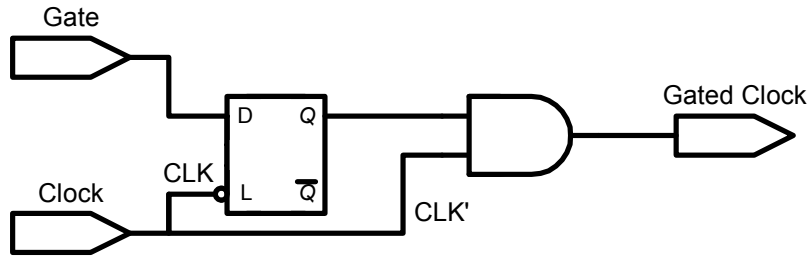


Figure 10: Latch Based Clock Gate Circuit in the Processors

(5.3) Realistic Cross-Talk Analysis

Note that the cross-talk effects on STA results are usually more severe using CeltIC runs without timing windows, as compared with the CeltIC runs with timing windows (shown in Figures 7, 8, and 9).

There are a few exceptions though. For example, under MaxTrans 1ns in Figure 7, the I/O ITagAddr[0] max_delay timing slack is -0.82ns using CeltIC with timing windows versus -0.78ns using CeltIC without timing windows. When CeltIC reads the TW file, it picks up the aggressor slews and victim slew from the file. In the absence of a TW file, CeltIC uses default values. Depending on the slews in the TW file (the victims may be switching very slowly, for instance) the push-out on the victim can change dramatically. Indeed for ITagAddr[0], the incremental SDF shows wider delay uncertainty after using the TW file, resulting in longer negative slack (i.e., -0.82ns):

ITagAddr[0] SDF without TW: (INTERCONNECT U24514/Y ITagAddr[0] (-45::61) (-20::25))
 ITagAddr[0] SDF with TW: (INTERCONNECT U24514/Y ITagAddr[0] (-41::64) (-30::50))

Across the three microprocessors used for this experiment, from simple to complex, there are a few to a dozen noise glitches having a sensitivity figure larger than one in the CeltIC runs without timing windows. For these same designs however, all the noise glitches generated from the CeltIC runs using timing windows have sensitivity figures of less than one. To ensure proper circuit operation, every restoring logic gate acted upon by a noise stimulus must have a time-domain DC-noise sensitivity ($dV_{\text{out}}/dV_{\text{in}}$) of less than one [2].

The two most sensitive nets in the experiment using the complex microprocessor with a DSP extension are plotted in Figure 11: V_{H} sensitivity -3.88 , V_{L} sensitivity -2.99 (without timing windows) vs. V_{H} sensitivity -0.63 , V_{L} sensitivity -0.52 (with timing windows). The glitch magnitude is also much reduced in the Celtic run with timing windows.

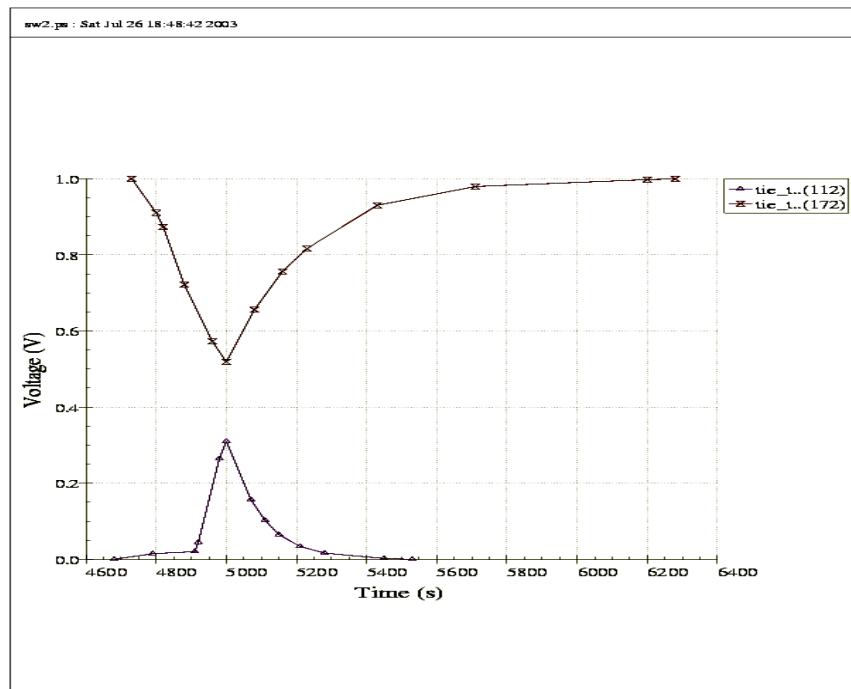
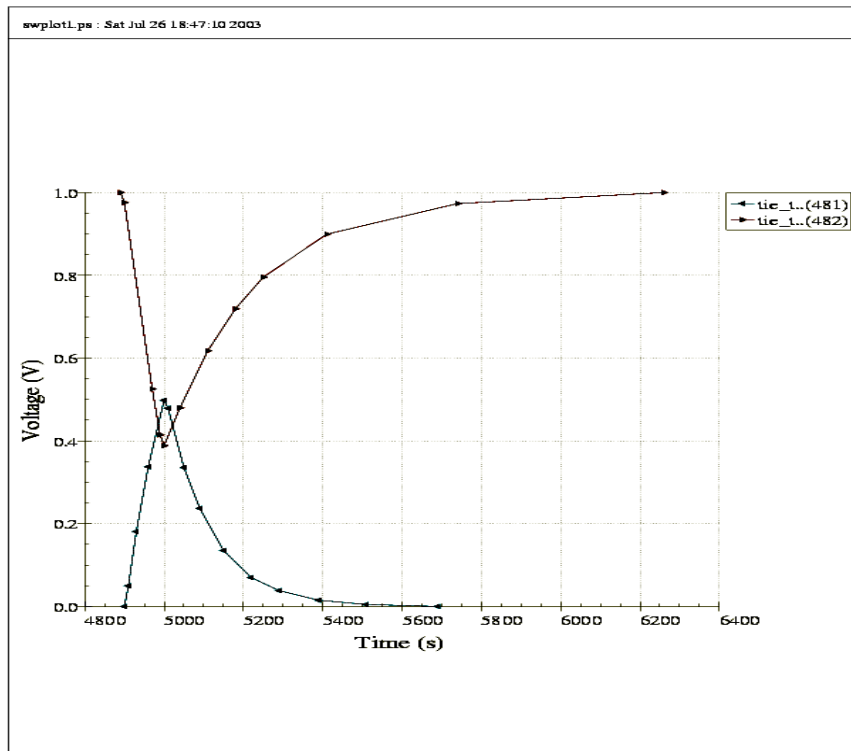


Figure 11: Top Two Most Sensitive Nets in Complex Microprocessor plus DSP Extension – without (top) and with (bottom) timing windows

Realistic cross-talk analysis using CeltIC should always use the timing window file with signal-arrival timing windows, slews, and slacks for the following two reasons:

- 1) To screen out false delays and glitch failures resulting from calculations based on logically impossible timing windows.
- 2) To use actual slew rates in both aggressors and victims to generate accurate incremental SDFs.

(5.4) Cross-talk Prevention and Fixing Techniques

Figures 7 and 8 show the impact of `set_max_transition_time` on STA results, which varies between designs and among path groups within the same design. Setting tight maximum transition constraints reduces the number of weak victim nets, however, fast transition constraints also mean strong aggressor nets. Thus there is an optimal maximum transition time for each design that results in minimal cross-talk impact on timing-critical path group of the entire design.

CeltIC can fix failures using the ECO option with Synopsys .lib files or with user-defined cell equivalents. CeltIC uses stronger drivers (cell swapping) to fix noise failures and outputs a new Verilog or DEF file, depending on which file was provided to the CeltIC run deck.

6. Conclusions

CeltIC can be integrated with existing P&R, 3D-extraction, and STA hardware-implementation tools. The resulting CeltIC signal integrity analysis and fixing flow has been demonstrated to generate files for signal-integrity ECO fixes for cross-talk-induced noise glitches and to perform SI-aware timing signoff.

The demonstrated SI analysis-and-fixing flow takes generic form SPEF with coupled RC generated from Fire & Ice, iterates between CeltIC and STA, and creates converged timing windows and incremental SDF for a final STA. CeltIC also outputs a new Verilog or DEF ECO file for the P&R tools to fix noise failures.

Running CeltIC-aided STA for various configurable and extensible Xtensa processors has shown the importance of setting maximum transition times during synthesis and P&R to minimize cross-talk effects on the design's critical timing paths.

7. Acknowledgement

The use of Artisan standard cell library (tsmc13lvfsg_sc-x_2003q1v1) and its CeltIC noise library (tsmc13lvfsg_sc-x-cdB_2003q3v1) in this work, is acknowledged.

8. Reference

- [1] ITRS 03 public conference, San Francisco, CA, July 2003
- [2] CeltIC User Guide, page 73, Product Version 4.2, Cadence, May 2003