Signal Integrity Management in an SoC Physical Design Flow

Murat Becer
Ravi Vaidyanathan
Chanhee Oh
Rajendran Panda
Motorola, Inc., Austin, TX

Presenter: Rajendran Panda

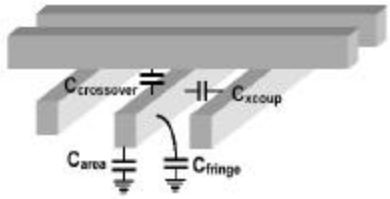
Talk Outline

- Functional and Delay Noise
- Correlation between them
- SI Methodology and Experiences
  - Preventive Measures
  - Functional Noise Repair
  - Delay Noise Repair
- Conclusions

Introduction

- Crosstalk noise is an undesired change in the voltage waveform of a net due to signal activity in its neighboring nets which are capacitively coupled to it.

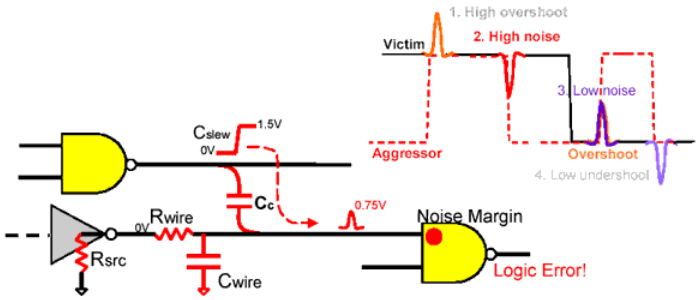
- Ratio of crosstalk capacitance to total capacitance is increasing.
- Faster slews result in increased injected noise
- More aggressive and less noise immune circuit structures are being used due to performance requirements.



Noise closure: a significant design and verification issue for large and high performance designs.

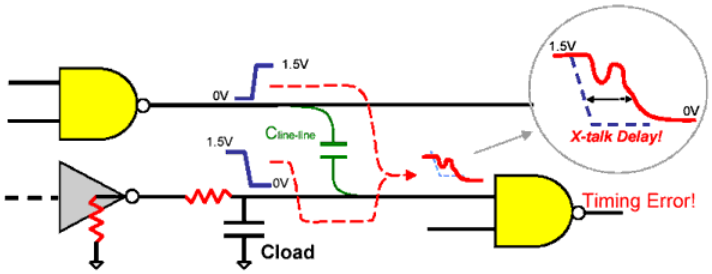
Functional Noise

- Crosstalk causes voltage glitches on quiet nets, resulting in false logic states being captured in the registers, causing functional failures.

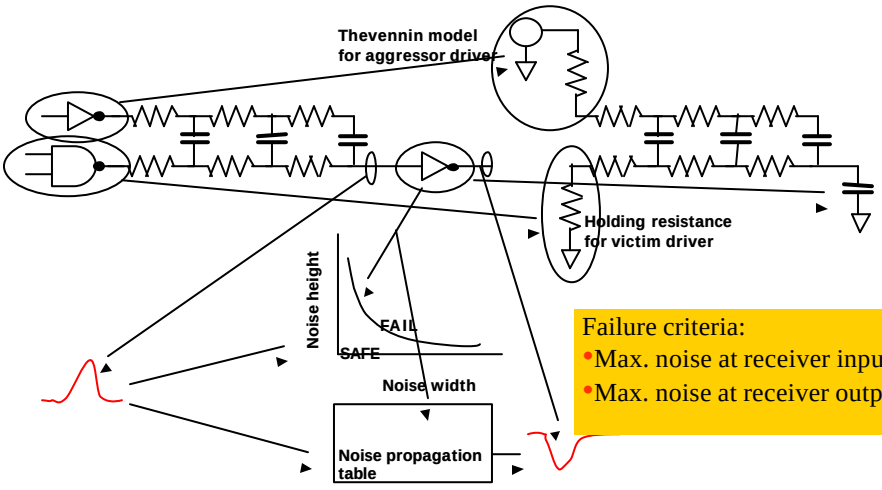


Noise on Delay

- Noise on delay changes the signal propagation on some of the nets, causing timing violations.

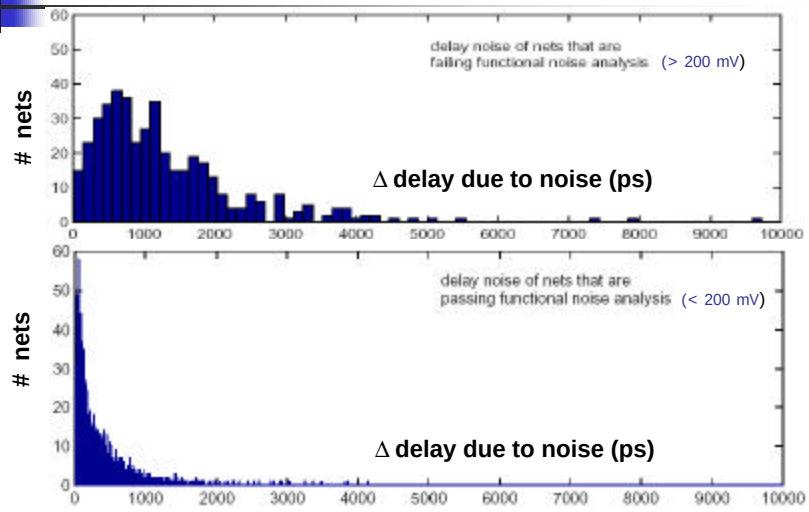


Noise Analysis



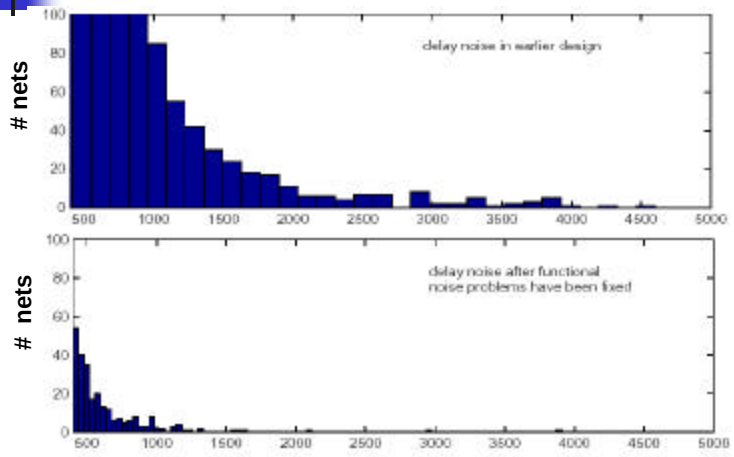
Failure criteria:
- Max. noise at receiver input
- Max. noise at receiver output

Functional and Delay Noise: Correlation



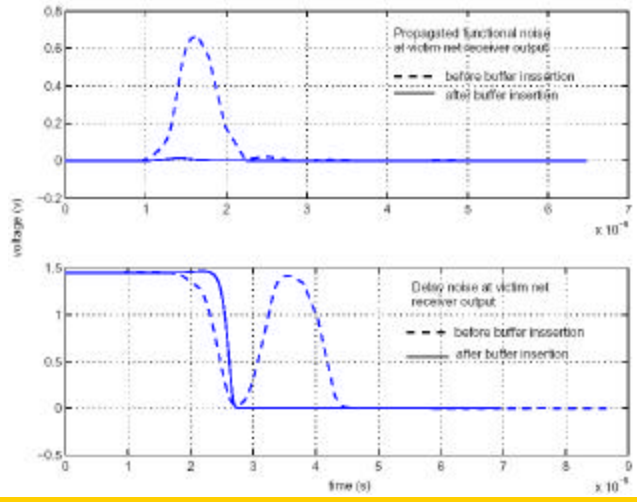
• Large functional noise also results in large delay noise

Functional and Delay Noise Correlation



• Fixing a functional violation often fixes many delay violations

Buffer Insertion

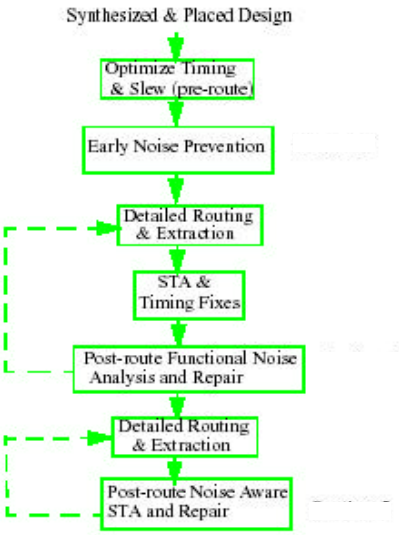


- Repair actions are effective for both functional and delay noise

Methodology Dilemma

- Design is SI clean only when all functional and delay violations due to noise are eliminated.
- Small glitches cause no problem functionally, but even small Δdelays of nets can add up to large path delays, suggesting delay problem is harder to deal with and so should be tackled early.
- On the other hand, delay noise analysis is inherently more expensive. Design groups tend to live with 'guard-banding' timing for noise and tackle only functional noise issues explicitly.
- There is a strong correlation in the occurrence and magnitudes of both these violations.
- Which violations should be tackled first? Delay violations or functional violations?

SI Closure Methodology



Why fix functional problems first?

- Since nets are shared by multiple timing paths, every path through a net failing functional noise criteria is likely to fail, even the non-critical ones.
  - Delay failure list is too large to manage efficiently, before functional noise fixes.
- Small glitches (which produce small delta delays) may not matter for many non-critical paths. So, after the large magnitude functional violators are eliminated, the number of violated paths decreases drastically.
- Easier to drive repair actions (buffering and sizing) using the noise magnitude metric (of functional noise), rather than the delta delay metric (of delay noise).

Case Study Details

- A wireless communication chip (SoC_Chip)
  - Integration with ~90K top level nets
  - 20 large SoC blocks/platforms
  - SoC blocks are delivered timing and SI clean
- A low-power IP platform (SoC_Platform)
  - 1 synthesized IP core, 11 synthesized modules, and 24 compiled memories
  - ~150K placed instances and ~160K nets
- Two functional blocks
  - ~45K nets SoC_Block_1
  - ~165K nets SoC_Block_2
- A high performance core (SoC_Core)
  - ~227K nets

All designs in 0.13um technology

Early Noise Prevention

- Preventive measures
  - Limit on parallel run length
  - Shielding of buses
  - Routing with extra spacing
  - Limit on slews
- Preventive actions do not require expensive noise analysis

Prevention: Parallel Run Limits

SoC Platform example:

	Limit on parallel run length	
	150um	300um
# delay violations	1936	3142
Worst delay slack	-0.22ns	-0.57ns

SoC Block-2 example:

	Limit on parallel run length	
	500um	No limit
# func violations	497	1013

• Few trial routes required to obtain a reasonable number

Prevention: Slew Constraints

SoC Platform example:

	Slowest transition time		
	0.4ns	0.7ns	1.0ns
#delay violations	2818	3258	5771
Worst delay slack	-0.22ns	-0.54ns	-0.70ns
#functional violations	91	171	177
#inserted buffers	3559	510	-

• Stronger victims overshadow the effect of sharper aggressor slews

• Increase in power consumption due to additional buffers is sub-linear due to reduction in short-circuit power

Prevention: Wide Spacing

SoC Platform example:

	Spacing	
	1x	2x
Delay noise on memory bus	0.22ns	0.01ns
Timing slack on path	-0.68ns	-0.02ns

Functional Noise Analysis

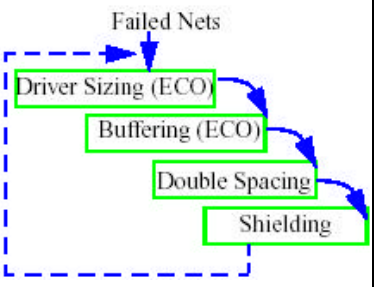
- Beware of false violations and false fixes
  - Lot of effort can get consumed with no real benefit
- Timing windows
  - Activity windows (for aggressors)
  - Sensitivity windows (for victims)
- Logic constraints
  - Invert, same, imply, set_high/low/stable
  - One-hot, one-cold, one-switching

For SoC Core Example	# Violations
No constraint	595
Logic constraints only	555
Activity windows only	187
Activity+Sensitivity windows	79
Logic + Timing constraints	50

Functional Noise Repair

Repair preferences:

- Sizing, buffering, and routing fixes suitable for block level noise repair
- Sizing not desirable at SoC integration stage
- Routing and buffering fixes preferred over sizing at SoC integration stage. (Assumes all SoC blocks are timing and SI clean.)



SI Convergence: Double Spacing

SoC Chip example:

Iteration	#violations remaining	#nets spaced	#violations fixed	Effectiveness (#fixed/#spaced)
1	885	240	356	1.48
2	529	176	198	1.13
3	331	254	220	0.87
4	111	308	64	0.21
Total	47	1078	838	0.78

- Law of diminishing returns applies to wide spacing

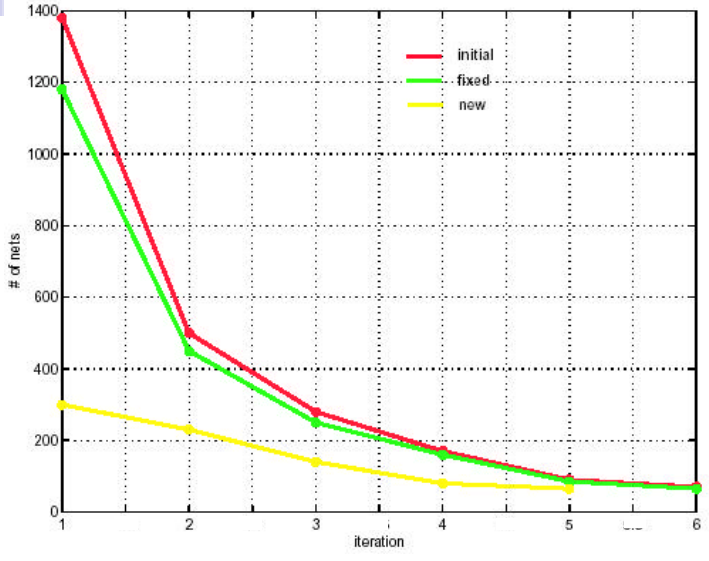
SI Convergence: Sizing and Buffering

SoC Block-1 example:

Iteration	# violations	# gates sized	# buffers inserted	# violations fixed
1	1380	1170	100	1180
2	500	400	80	450
3	280	200	50	250
4	170	120	45	160
5	90	75	40	85
6	70	50	15	65

- Spacing not attempted because design was congested and only 4 metals were available

SI Convergence: Sizing and Buffering



Size First or Buffer First?

Iteration	Sizing before buffering			Buffering before sizing		
	#Violations	# Sized	# Buffered	#Violations	# Buffered	# Sized
1	208	197	7	208	201	2
2	51	48	1	76	72	1
3	23	19	1	48	43	0
4	11	8	0	30	26	0
5	10			22		
Total		272	9		342	3

• Sizing is less intrusive, causes less disruption to routing, and is easily implemented with incremental routing

Timing with Noise

SoC Platform example:

	# Setup Violations	Timing Slack
STA with 2.0X Coupling Multiplier	4865	-1300 ps
STA with Delay Noise	2936	-620 ps
STA with 1.5X Coupling Multiplier	1138	-560 ps

• STA with delay noise is slow due to the added noise analysis overhead and iterations for timing windows convergence

• Assume infinite windows for first iteration and iterate over critical paths only, for faster analysis

Delay Noise Repair

1. STA with coupling and infinite windows
2. STA iterations with coupling and updated windows
3. For each failing path
    1. Select a net contributing most delta delay
        1. Resize driver to next higher size
        2. Incremental STA
        3. If new timing violations on other paths
            1. Revert back to original size
            2. Create router constraint
        4. Else if timing of path improves and no new timing violations
            1. Accept and legalize placement
            2. If path meets timing now, take up next path (Step 3)
            3. If not, take up next net in the path (Step 3.1)

Delay Noise Repair

- SoC Platform example:
    - Num. STA iterations with windows: 2
    - Initial violations: 88 (setup), 63 (hold)
    - Initial slack: -100ps (setup), -20ps (hold)
    - Num. Drivers sized: 76
    - Num. Nets spaced: 12

Summary

- Occurrence and magnitude of functional violations and delay violations in a design are highly correlated.
- It is efficient to tackle functional violations first, and then deal with delay violations.
- Preventive actions are very valuable. Moreover, they do not require noise analysis.
- Sizing fixes are less intrusive than buffering, and gives quicker SI convergence.