# Soft-Spot Analysis: Targeting Compound Noise Effects in Nanometer Circuits

**Chong Zhao and Sujit Dey**
University of California, San Diego

**Xiaoliang Bai**
Cadence Design Systems

Soft-spot analysis identifies regions in a circuit that are most susceptible to multiple noise sources and their compound effects so that designers can harden those spots for greater robustness. HSpice simulation validates the methodology's quality, and demonstration on a commercial embedded processor shows its scalability.

**According to** the 2001 *International Technology Roadmap for Semiconductors* (http://public.itrs.net/Files/2001ITRS/Home.htm), with feature sizes shrinking to nanometer scale, clock frequencies reaching the multi-GHz level, and supply voltages declining to the subvoltage range, effects of various noise sources are becoming stronger than ever. Meanwhile, semiconductor device noise margins are shrinking significantly. As a result, nanometer circuits are becoming more vulnerable to signal integrity issues such as crosstalk and *IR* drop, as well as radiation-induced transient soft errors. Even worse, as this article demonstrates, different noise sources and failure mechanisms tend to interact, creating compound effects that exacerbate the difficulty in analyzing and designing reliable digital-circuit systems.

Significant research and technology developments aim to ensure the reliability of nanometer chips. Researchers have widely investigated analysis and optimization techniques for signal integrity issues such as crosstalk,[1] *IR* drop,[2] ground bounce,[3] and substrate noise[4] in SoCs. In addition, the effects of process variations[5] and manufacturing defects have drawn extensive research interest because large uncertainties in device parameters can lead to yield and performance degradation. However, most noise analysis and prevention techniques apply to a single noise source and therefore cannot address the evolving reality of multiple interacting noise sources. Moreover, even single-noise analysis techniques are computationally intensive, and the lack of scalability limits their applicability to complex SoCs.

Nanometer circuits are also becoming more vulnerable to radiation effects and other sources of soft errors. Single-event upsets (SEUs) caused by cosmic ray neutrons or alpha particles[6] severely impact field-level product reliability. Simulation-based methods adopted to analyze SEUs (IROC Technologies, http://www.iroctech.com) are prohibitively time-consuming. Researchers have developed concurrent error detection and concurrent circuit-hardening techniques to protect circuits from SEUs.[7,8] Although they offer feasible solutions, these techniques require significant engineering effort and design overhead. Recently, Mohanram and Touba suggested partial protection,[9] but an efficient methodology for evaluating circuit robustness wasn't developed. Therefore, the selection of spots to be hardened might not be optimal, and a design's most vulnerable region might not be protected. The result is an insufficient partial protection solution.

Despite extensive research on individual noise sources, few researchers have tried to develop analysis techniques for combined noise and soft-error effects. As this article demonstrates, different noise sources tend to aggravate one another's effects, making stand-alone noise analysis results inaccurate. At the same time, the unpredictable and transient nature of these noise effects makes online detection and protection schemes inevitable, because design and deterministic manufacturing testing alone can no longer guarantee 100% robustness. However, blindly applying hardening techniques to the entire design incurs unacceptable design overhead. Finally, as the circuit size drastically increases, simulation-based approaches cannot complete within a reasonable time. Therefore, a methodology that statically evaluates the impact of compound noise effects on nanometer circuits and identifies the vulnerable regions is essential for efficient and economical robust digital circuit design.

To overcome the challenges just cited, we propose an efficient soft-spot analysis methodology aimed at compound noise effects. Fundamentally different from traditional approaches that focus on the behaviors of random and transient noise interference, our approach targets a design's noise immunity, an intrinsic circuit characteristic that doesn't depend on external noise interferences but is closely related to the timing, logic, and electrical features of the design that can be conveniently analyzed during the early design phase. Instead of trying to predict how and when different noise effects will occur without enough information about the unpredictable sources, the proposed methodology evaluates the probability that noise occurring at different nodes in the circuit will cause a system malfunction. This analysis is based primarily on the circuit's structural information and is applicable even without specific knowledge of noise sources. Moreover, any additional useful information about various noise sources can enhance analysis accuracy.

Our methodology provides an overall vulnerability distribution and reveals that the inherent noise tolerance of different circuit regions vary greatly. Designers can then further investigate the most vulnerable nodes through focused noise analysis, eliminating potential noise effects through limited design modifications and selective application of online hardening techniques. As a result, the cost and effort of designing highly robust circuits can shrink dramatically. Because our methodology is a static approach that doesn't require dynamic simulation or intensive computation, it can handle large, complex circuits. Numerical results show that the proposed methodology is accurate, efficient, and scalable to large and complex designs.

## Compound effects of multiple noise sources

Various physical mechanisms cause multiple noise sources in nanometer circuits. Among the noise sources that might coexist in a nanometer circuit are

- crosstalk caused by signals switching on strongly coupled wires,
- *IR* drop and ground bounce caused by excessive simultaneous current draw from the resistive/inductive power grid,
- substrate coupling noise, and
- environmental variations such as soft errors induced by particle strikes during chip operation.

Each of these effects can cause circuit failures; furthermore, as our simulations show, different noise sources can combine to magnify their effects, greatly increasing the possibility of errors.

In the circuit shown in Figure 1, because of coupling capacitances $C_x1$ and $C_x2$ between the victim net (in the middle) and the two aggressor nets, when the input of INV1 remains at 0, a $0 \rightarrow 1$ transition at the input of INV3 or INV5 will result in a negative crosstalk glitch on the victim net, which will propagate to the input of a D-type flip-flop (DFFV) through INV2. However, as shown by INV2 input voltage $V$(xv) and output voltage $V$(dv) in Figure 2a, even in the worst case when both aggressors switch simultaneously in the same direction, the output glitch of INV2 is not strong enough to be captured as a stable logic error in the DFFV, so the DFFV remains at the correct value—0 [$V$(qv) in Figure 2a]. However, if a particle strikes the sensitive region of either INV1 (Figure 2b) or INV2 (Figure 2c) at a certain time, with all the other conditions unchanged, an erroneous 1 is latched in the DFFV. Our experiment models soft errors as current sources[10] between the drain of the MOS transistor and the output node. In another case, the same crosstalk glitch that wasn't strong enough to change the state of DFFV is turned into a latched error by an *IR* drop in the DFFV power line (Figure 2d). In all the failure cases, the error effects are the same: an observable error captured by the DFFV.

These experiments show that although the essential physical mechanisms of these noise effects differ, they can affect circuit behavior in a combined manner. A single noise source that isn't strong enough to affect behavior might be intensified by other noise effects. Therefore, a system exposed to multiple noise sources becomes more vulnerable. Furthermore, simply examining the erroneous behavior won't pinpoint the noise source(s) causing the failure. Hence, methodologies that try to address the effect of a single noise source while ignoring other sources might be both overly optimistic and inefficient. We propose a promising solution that considers multiple noise sources and efficiently evaluates a given design's overall vulnerability.

## Soft-spot-analysis methodology

Random occurrences and complex physical mechanisms of noise and their interactions depend on many

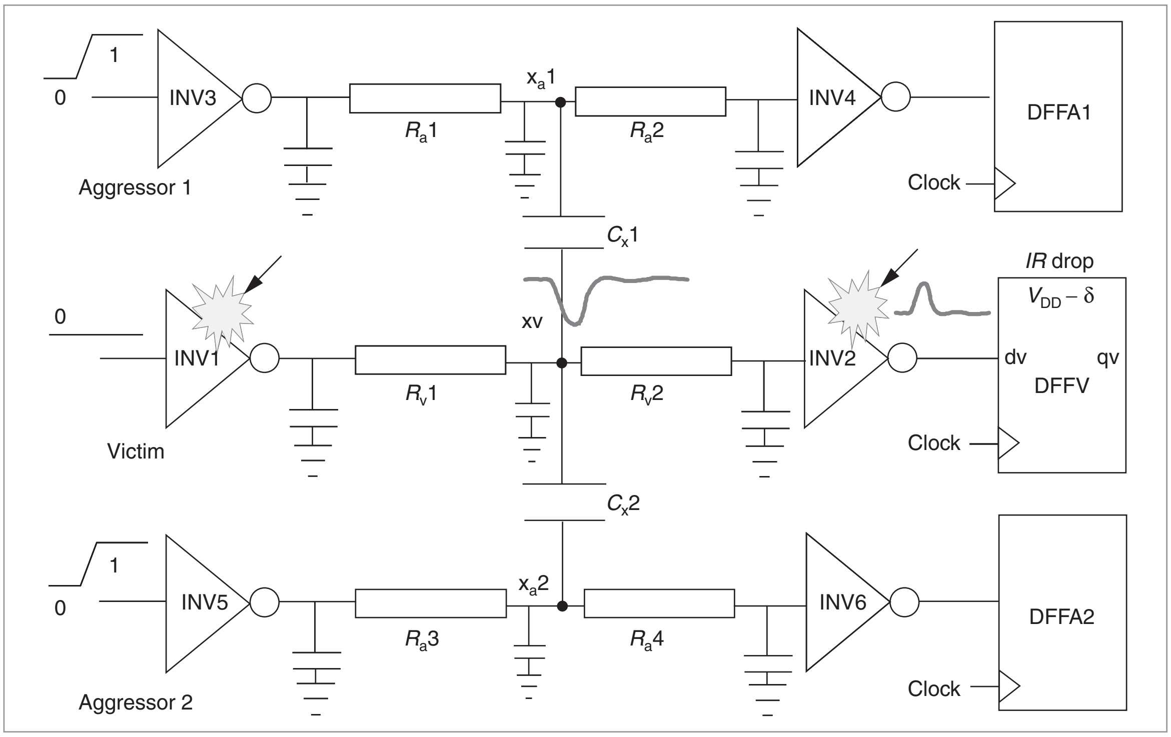


**Figure 1. Example circuit showing compound noise effects.**

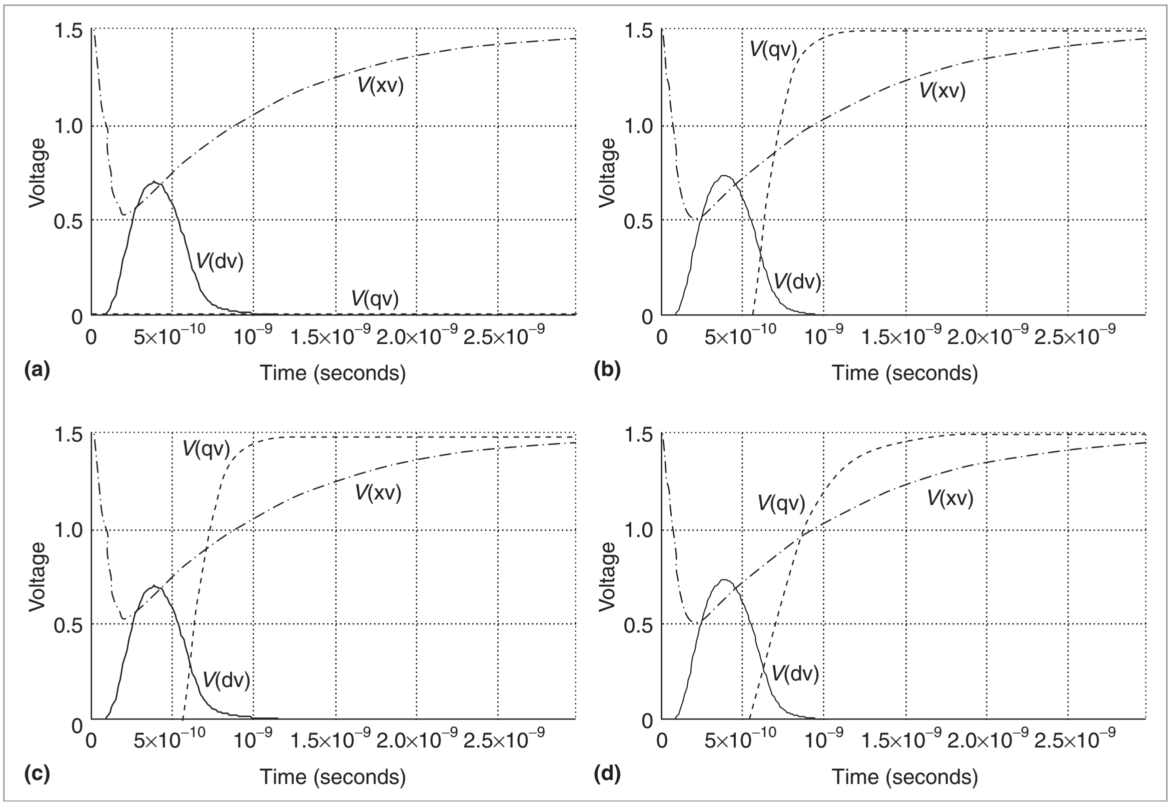


**Figure 2. HSpice simulation of single and compound noise effects: crosstalk only (a); crosstalk and a particle strike on INV1 (b); crosstalk and a particle strike on INV2 (c); crosstalk and an *IR* drop in the DFFV power line (d).**

factors that cannot be precisely determined until the product is manufactured and operating in a real environment. As a result, the behaviors and aggregated effects of various noise sources are extremely difficult to predict during chip design. However, using various EDA tools, designers can effectively analyze information about the design, such as timing feature, logic paths, and layout-extracted electrical characteristics, during the design phase. Together, these factors influence the design's noise immunity, an intrinsic characteristic that is independent of the external noise disturbances. By studying these design features, we can estimate the ability of different regions in a design to resist potential noise interferences, and we can predict the severity of functional impact if a noise occurs. These ideas form the basis of our soft-spot-analysis methodology.

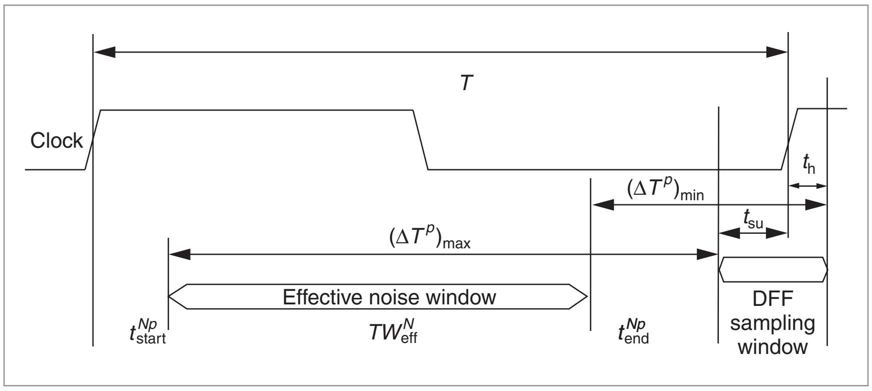


**Figure 3. Calculating the effective noise window.**

For each node $N$ in a given digital circuit, we define the softness $S_N$ as its vulnerability to noise, reflected by the node's tendency to allow noise to propagate through it with enough strength and proper timing to eventually cause observable errors. An observable error is one that is latched into a memory element and thus becomes a stable erroneous logic value. The objective of our soft-spot analysis is therefore to determine the magnitude of $S_N$ for all circuit nodes and to identify a collection of soft spots as the nodes with high softness values.

Not all noise occurring inside a digital circuit can eventually cause functional errors. Three well-known masking effects—timing masking, electrical masking, and logic masking—all tend to prevent a noise from causing observable errors. Correspondingly, $S_N$ should reflect all three masking effects at a circuit node. First we introduce novel methods to obtain numerical representations of all three masking effects, and then we calculate $S_N$ as a function of the strengths of these factors.

Our current work focuses only on the glitch-type noise, which we model as an electrical pulse with certain magnitude and duration. A little additional effort and further research will let us consider the delay-type noise as well.

## Timing masking

Timing masking means that noise can cause an observable error only if it is captured by a memory element. To be captured, it must arrive at the memory element's input within a sampling window. For a DFF, the sampling window is bounded by setup time $t_{su}$ and hold time $t_h$ around the active clock edge, as shown on the right side of Figure 3. To determine the required time interval for noise at a node to reach a DFF within its sampling window, we define the effective noise window $TW^N_{eff}$ such that only noise existing at node $N$ overlapping with $TW^N_{eff}$ can reach at least one DFF during the DFF's sampling window. In other words, if a noise originates or arrives at node $N$ before the start (or after the end) of $TW^N_{eff}$, it will reach all DFFs before the start (or after the end) of their sampling window and will therefore not be captured by any DFF. As Figure 3 shows, the $TW^N_{eff}$ of a specific path ($p$) is bounded by start time $t^{Np}_{start}$ and end time $t^{Np}_{end}$, determined by the worst-case longest delay $(\Delta T^p)_{max}$ and best-case shortest delay $(\Delta T^p)_{min}$ from $N$ to the DFF through $p$, respectively. If the clock period is $T$, it is easy to see that $t^{Np}_{start} = T - t_{su} - (\Delta T^p)_{max}$ and $t^{Np}_{end} = T + t_h - (\Delta T^p)_{min}$.

Because there are usually multiple DFFs reachable from node $N$ through many logic paths, we use the maximum (latest) $t^{Np}_{end}$ and the minimum (earliest) $t^{Np}_{start}$ among all paths to calculate $TW^N_{eff}$. Let $P$ be the collection of all possible paths through node $N$:

$$TW^N_{eff} = \max_{p\in P}\left\{t^{Np}_{end}\right\} - \min_{p\in P}\left\{t^{Np}_{start}\right\} \quad (1)$$

This $TW^N_{eff}$ gives a pessimistic timing requirement for noise occurrences at node $N$ and provides a measurement of the timing masking effect's strength: The larger a node's timing window, the more likely it is that noise at this node will overcome the timing masking effect. This timing window requirement is an efficient measurement of the timing characteristic of glitch-type noise propagation. It's possible to model the delay effects of noise as variations in the derived effective noise window, but this is beyond the scope of this work.

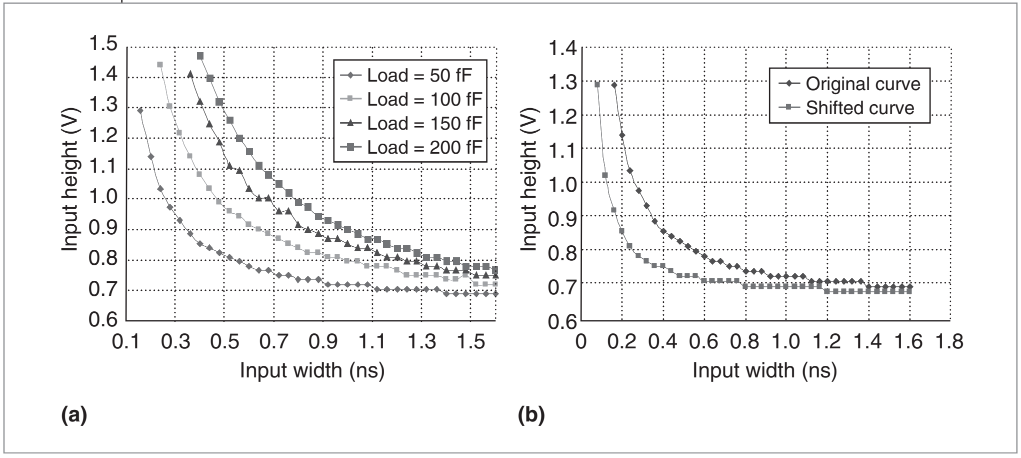


**Figure 4. Noise rejection curves and curve shift: noise rejection curves of an inverter under varying capacitive loads (a); curve shift resulting from crosstalk (b).**

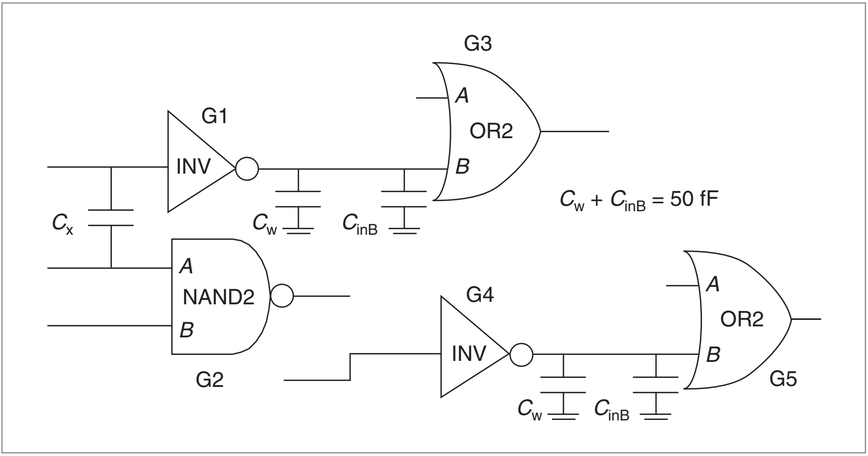


**Figure 5. Example circuit with crosstalk to demonstrate curve shift.**

## Electrical masking

Electrical masking means that noise must have enough duration and amplitude to propagate through multiple logic gates. We can represent the strength of a single gate's electrical masking effect by the gate's noise rejection curves (NRCs). Figure 4a shows an example of NRCs for an inverter in a 0.18-micron cell library with different capacitive loads. The *x*- and *y*-axes are the input noise's width and height. The curve is such that a glitch can propagate through the gate with enough strength (characterized by a predefined threshold voltage) only if its shape is in the region above the curve (the noise-sensitive region). The region under the curve is the noise-immune region. The NRC is also a function of the capacitive load driven by the gate. With all other conditions unchanged, the NRC is closer to the axes when driving a smaller load, indicating that it is more vulnerable than the same gate driving a greater load capacitance.

We can view an NRC as a representation of a gate's immunity to noise caused by arbitrary noise sources. In reality, the nature of certain noise sources, such as radiation effects, might be completely unknown until field operation, whereas the effects of other noise sources, such as crosstalk, can be estimated on the basis of the circuit's *RC* characteristics. If we can insert information about those analyzable noise sources into the NRC, we can use the modified curves to measure the remaining noise margin owing to those unpredictable noise sources. We realize this idea through the concept of a *curve shift*.

The circuit shown in Figure 5 illustrates the curve-shift idea using crosstalk as an example. Gates G1 and G4 are identical inverters with the same 50-fF load capacitance, including the input capacitance of pin B of the two-input OR gate G3 ($C_{inB}$) and the wire capacitance ($C_w$). They should have the same 50-fF NRC shown in Figure 4a. However, if a coupling capacitance ($C_x$) exists between the inputs of G1 and G2, G1 will, under certain circumstances, experience crosstalk noise at its input and become less resistant than G4 to any other random noise disturbances in addition to this crosstalk that might occur at its inputs. Therefore, using the same NRC to describe the noise tolerance of G1 and G4 becomes inappropriate. Some glitches in the original NRC's noise-immune region might propagate through G1, meaning they should be in the noise-sensitive region. We can also see this as an expansion of the noise-sensitive region from the original NRC by shifting the curve toward the axes, as shown in Figure 4b.

A quick way to determine the amount of the shift is to estimate the worst-case magnitude of the crosstalk using existing techniques (for example, the extended 2-$\pi$ model[11]) and then downshift the curve toward the *x*-

axis by this amount. It's possible to apply this method during the soft-spot analysis, but because it considers only the worst case, the result is pessimistic. A more sophisticated, but also more time-consuming, way is to create the shifted curves under certain crosstalk scenarios using offline simulations (as we did to generate Figure 4b) and store the curves in a database along with the original NRCs. During circuit noise analysis, a proper curve is retrieved from the database on the basis of the loading conditions and the estimated crosstalk strength.

Now, in the NRC graph, we define the noise propagation ratio $R_e^N$ as

$$R_e^N = \left( \frac{A_{\text{sen}}}{A_{\text{imm}}} \right)_{\text{NRC}} \tag{2}$$

where $A_{\text{sen}}$ is the area of the noise-sensitive region and $A_{\text{imm}}$ is the area of the noise-immune region. When calculating the areas, we set the upper bound on the $y$-axis to be the maximum possible input glitch height, which is the power supply voltage, and we set the upper bound to be the maximum possible input glitch width, which we assume to be the clock period. $R_e^N$ lets us measure the strength of the electrical masking effect at node $N$: A higher $R_e^N$ means a larger noise-sensitive region; therefore, more glitches can propagate through the gate, and the node is more vulnerable.

Although noise propagates in a circuit, when evaluating a single node's electrical masking effect, it isn't necessary to consider the electrical masking of other nodes, including those on the logic paths en route to the primary outputs (POs). We can understand this by looking at both the conceptual and technical aspects.

Conceptually, the essential idea of soft-spot analysis is to evaluate how an individual circuit node can effectively contribute to prevent a glitch-type noise from producing observable errors. It doesn't mean studying the noise-propagating behavior. A stronger electrical masking effect at a single node will tend to reduce the possibility that an incoming glitch will propagate with enough strength, and this makes the glitch less harmful to the subsequent logic. From this point of view, the electrical masking effect is localized to the node, and the designer needn't consider the electrical masking of the subsequent gates.

Technically, because of the high gains of digital CMOS logic gates in a transition region,[12] a small change in the input voltage produces a large output variation in the region. As a result, when a glitch reaches a gate's input, if its shape falls in the noise-sensitive region—even if it's only slightly above the curve—the output glitch will become large enough to propagate through the remaining gates (the subsequent gate will not be able to electrically mask it) on its path to a PO. On the other hand, if it falls in the noise-immune region, even if it's just slightly below the curve, the output glitch will be so small that it will not propagate through additional gates. This means its shape will not gradually change as it propagates through a chain of logic gates. Instead, the electrical characteristic of the first gate that it encounters almost completely determines whether it can electrically propagate to the POs. Under certain boundary conditions (when the voltage gain is close to unity and the output glitch's shape resembles that of the input glitch), designers should consider the electrical masking effects of the subsequent gates. However, the probability that this boundary condition will persist over several levels of logic gates is very low. Taking a first-order approximation, we don't consider the effect of the gates en route to the POs.

Here we describe how to calculate $R_e^N$ for every circuit node during circuit noise analysis. For cell-based designs, we first precalibrate the standard-cell library using offline HSpice to create an NRC database of all gates with different capacitive loads. Then, for a given design, we obtain the load capacitance of each node from layout-extracted *RC* information to retrieve the proper curve from the database. This retrieved curve can then be shifted according to preliminary analysis results of certain noise sources. For example, given the detailed *RC* parasitic and coupling capacitances, we can estimate crosstalk effects using existing techniques such as the extended 2-π model.[11] Finally, we compute $R_e^N$ as the area ratio in the modified curve. Figure 6 illustrates this process. Given specific design and process information, we can similarly consider curve shifts caused by other mechanisms.

An obvious advantage in the flow just described is that one-time cell calibration is sufficient for a given library and can apply to all circuits implemented in the same library. As a result, we can make full use of HSpice's accuracy without repeatedly suffering from its time-consuming nature. Another advantage is that we build noise information into the single entity $R_e^N$ by applying a curve shift; therefore, using existing noise analysis techniques can improve accuracy.

### Logic masking

Logic masking refers to the effect that noise ceases to propagate through a gate whose output is solely determined by inputs other than the one carrying the noise. The chances that noises occurring at different nodes will

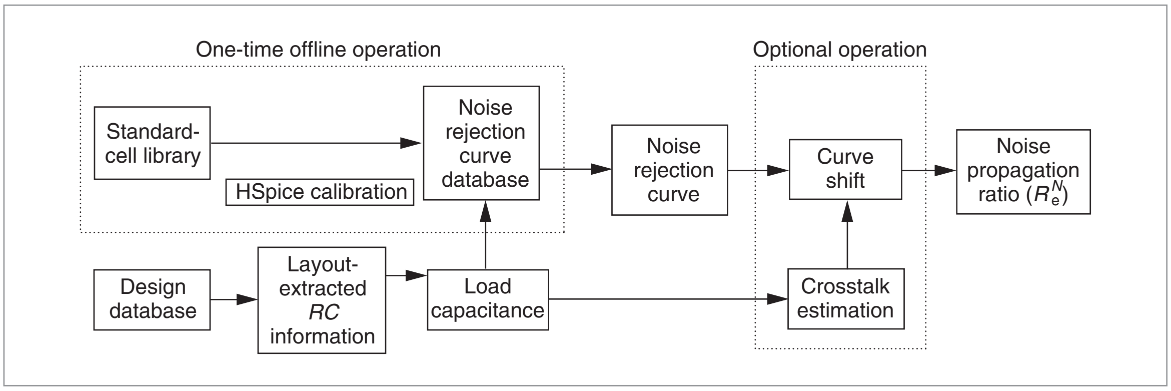
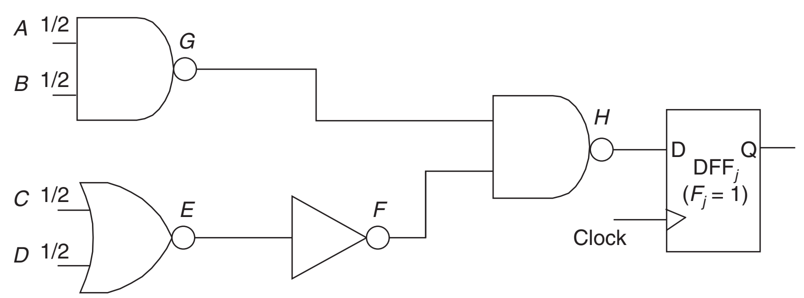


**Figure 6. Calculating noise propagation ratio $R_e^N$ in cell-based designs.**

Step 1: Forward breadth-first search

- $Pr^E(1) = Pr^C(0) \times Pr^D(0) = 1/4$
  $Pr^G(1) = 1 - Pr^G(0) = 1 - [Pr^A(1) \times Pr^B(1)] = 3/4$
- $Pr^F(1) = Pr^E(0) = 1 - Pr^E(1) = 3/4$
- $Pr^H(1) = 1 - Pr^H(0) = 1 - [Pr^F(1) \times Pr^G(1)] = 7/16$



Step 2: Backward breadth-first search

- $P^F_{\text{prop}} = Pr^G(1) \times P^H_{\text{prop}} = 3/4$
  $P^G_{\text{prop}} = Pr^F(1) \times P^H_{\text{prop}} = 3/4$
- $P^E_{\text{prop}} = P^F_{\text{prop}} = 3/4$
  $P^C_{\text{prop}} = P^E_{\text{prop}} \times Pr^D(0) = (3/4)(1/2) = 3/8$, $P^D_{\text{prop}} = \ldots = 3/8$
- $P^A_{\text{prop}} = P^G_{\text{prop}} \times Pr^B(1) = (3/4)(1/2) = 3/8$, $P^B_{\text{prop}} = \ldots = 3/8$

**Figure 7. Example circuit to demonstrate calculation of logic masking factor.**

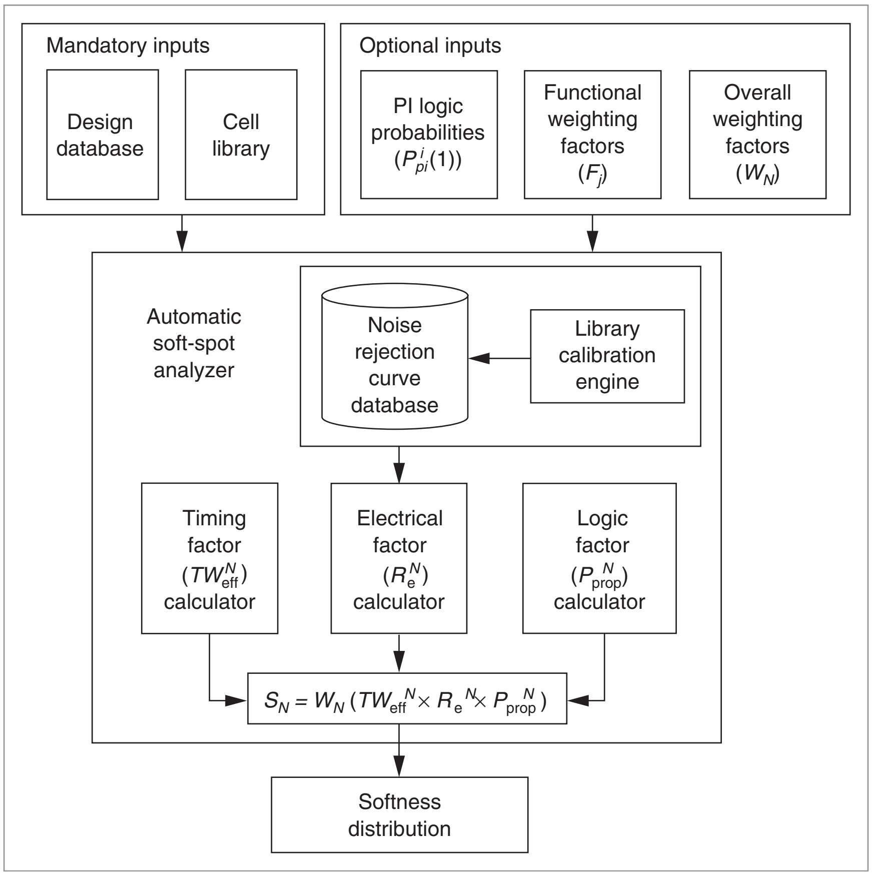
survive multiple levels of logic gates and eventually reach the memory elements depend on the logic structure. Complete determination of the logic masking effect requires exhaustive exploration of the entire input vector space and prohibitively long dynamic simulation time. As an alternative, we developed an efficient logic-path tracing algorithm to estimate the propagation probability ($P^N_{\text{prop}}$), defined as the ability of a glitch propagating from node *N* to extend to all reachable DFFs through legitimate logic paths. According to this definition, the input of a DFF with no other fan-out has a propagation probability of 1, indicating that a glitch at the DFF's input will be able to logically reach one DFF.

Our algorithm has two steps, each using the breadth-first search (BFS) algorithm to go through the design's gate-level netlist.[13] The first step uses a forward BFS, starting from the primary inputs (PIs), to derive for each node the logic probability—that is, the probability of being logic 1 (or 0)—denoted by $Pr^N(1)$, $Pr^N(0) = 1 - Pr^N(1)$. Figure 7 best illustrates how to calculate the logic probabilities of all nodes. Assuming the logic probabilities for all inputs [$Pr^A(1)$, $Pr^B(1)$, $Pr^C(1)$, and $Pr^D(1)$] are 1/2, the algorithm calculates $Pr^E(1)$, $Pr^G(1)$, $Pr^F(1)$, and $Pr^H(1)$ in order as the netlist is searched, and the calculated results are listed above the circuit. To start the process, the logic probabilities at the PIs should be known; these are usually available through functional vector simulations. More specifically, designers can obtain them by recording the statistics of logic 0s and 1s applied to the PIs during functional verification. If such information isn't available, assuming that logic 1 and 0 have equal probabilities at all PIs is a good approximation.

The second step uses a backward BFS, starting from the input nodes of the DFFs, to calculate $P^N_{\text{prop}}$ at each node. As the algorithm searches the netlist backward, it calculates the value at an input of a gate *M* (a descendant node) from the value at the output node of gate *M* (the parent node) and the probability of all the side inputs carrying noncontrolling values of gate *M*. The algorithm determines this probability from the logic probabilities at the side inputs obtained during the forward BFS. In the same example in Figure 7, $P^H_{\text{prop}}$ is set to 1 because node

*H* is the input of a single DFF. Next, the search reaches nodes *F* and *G* during the backward BFS. The algorithm calculates $P^F_{\text{prop}}$ as $P^H_{\text{prop}}$ multiplied by $Pr^G(1)$, because the noncontrolling value of a NAND gate is 1. The algorithm similarly calculates $P^G_{\text{prop}}$ at the same time. As the search continues, the algorithm calculates the values at nodes *E*, *A*, and *B* and then nodes *C* and *D* in turn. The calculations and results appear under the circuit in Figure 7. If a node has multiple parents (multiple fanouts), the total of the values derived from all parents gives the cumulative propagation probability at the node.

In reality, a design's DFFs might not all be of equal functional significance. Designers might assign the *j*th DFF a functional weighting factor $F_j$ on the basis of design-specific knowledge. For example, a DFF that stores a crucial control signal such as global reset, clock gating enable, or interrupt status should be assigned a higher weight, indicating that an observable error latched in this DFF will have a greater functional impact. As a result, the propagation probability of the input to the *j*th DFF will be equal to $F_j$, and the backward BFS will start from different DFFs with different weights. Hence, the propagation probability of each node obtained thereafter contains extra information about the functional impact if a glitch were to occur at the node.



Mandatory inputs
Design database
Cell library
Optional inputs
PI logic probabilities ($P^i_{pi}(1)$)
Functional weighting factors ($F_j$)
Overall weighting factors ($W_N$)
Automatic soft-spot analyzer
Noise rejection curve database
Library calibration engine
Timing factor ($TW^N_{\text{eff}}$) calculator
Electrical factor ($R^N_e$) calculator
Logic factor ($P^N_{\text{prop}}$) calculator
$S_N = W_N (TW_{\text{eff}}^N \times R_e^N \times P_{\text{prop}}^N)$
Softness distribution

**Figure 8. Automatic soft-spot analyzer.**

## Evaluating softness and identifying soft spots

We have described ways to measure the three masking effects. Softness $S_N$ should be a function of the timing factor $TW^N_{\text{eff}}$, the electrical factor $R^N_e$, and the logic factor $P^N_{\text{prop}}$. Although $S_N$ might have many possible analytical forms, if we consider $TW^N_{\text{eff}}$, $R^N_e$, and $P^N_{\text{prop}}$ to contribute independently, we can express $S_N$ as

$$S_N = W_N (TW^N_{\text{eff}} \times R^N_e \times P^N_{\text{prop}}) \tag{3}$$

In Equation 3, $W_N$ is an optional application-specific weighting factor at node *N*, with 1 as the default, for designers to convey design-related knowledge. Examples of information contained in $W_N$ include a preliminary analysis result of temperature variations across the chip during operation, empirical data about process variation, or even information about a potential radiation hit. This weighting factor gives the proposed methodology additional flexibility and controllability.

We developed an automated flow called the automatic soft-spot analyzer (ASSA), shown in Figure 8, to implement the methodology. To execute, ASSA first uses a library calibration engine to generate an NRC database for the cell library. This step is a one-time operation for each library, after which ASSA can read in the database from storage when analyzing a given design. Next, ASSA uses information in the design database (gate-level netlist, timing, physical layout, extracted *RC*, and so on) to evaluate timing, electrical, and logic factors. ASSA then calculates $S_N$ and provides a softness distribution as its output. From this distribution, we can identify a set of soft spots as nodes with high softness values. In addition, optional inputs, including PI logic probabilities, DFF functional weighting factors, and overall weighting factors, might also be provided to improve the results' accuracy and validity by giving the tool more information about the circuit functionality and the external environment.

Selection of soft spots according to the softness distribution relates closely to the affordable design cost. A cost metric can help in determining the softness value threshold, $S_{th}$: A lower threshold means that more nodes will be identified as soft and that the cost to analyze, revise, and protect these soft spots will be greater, resulting in a higher level of robustness. For mission-critical applications such as medical equipment, $S_{th}$ should be set low so that ASSA marks a larger portion of nodes as soft spots for designers to make the design highly noise immune at a higher cost. For cost-sensitive commercial applications, $S_{th}$ should be set higher so that designers need consider further robustness optimization only for the nodes most likely to cause the largest functional impact if affected. Determining the cost metric and robustness optimization are topics of ongoing research beyond the scope of this work.

### Methodology limitations

Soft-spot analysis not only provides an efficient method for quickly identifying a circuit's most vulnerable spots but also proposes a new viewpoint on circuit reliability analysis. However, the current methodology has some limitations. Equation 3 assumes that the timing factor, the electrical factor, and the logic factor are independent of each other. However, this assumption is oversimplified, because contributions by each of the three factors might be affected by the others. To determine overall softness, instead of a simple equation we need an empirical metric that considers the interaction among the three factors.

The algorithm for calculating the logic masking effect doesn't apply to reconverging paths. Complete determination of the logic probability is a circuit-satisfiability problem and has been proven to be NP-complete.[13] Much work has been done attempting to find approximation algorithms.[14] The algorithm we used is fast, efficient, and has a limited loss of accuracy. Moreover, its accuracy can improve with the use of an existing algorithm. (BFS is neither the only choice nor the best choice.) However, the well-studied satisfiability problem is not the main focus of this work.

The current methodology doesn't consider the effects of process variations or manufacturing defects. Process variation is becoming a major uncertainty in circuit reliability, and researchers can best describe its effects probabilistically. Improved soft-spot analysis would incorporate distributions of the three factors on the basis of process information.

## Experimental results

The proposed methodology can accurately identify the most vulnerable nodes in a circuit and is useful for large systems, because its runtime is almost linear to the number of circuit nodes.

### Accuracy and efficiency

We applied the proposed methodology to four circuits to evaluate its quality and speed by comparing our results with accurate HSpice fault simulation results. Because of HSpice's speed limitation, we can perform simulation only on small circuits. Two of the four circuits are basic blocks in many digital designs (adder, which is a 4-bit adder; and DEC, which is a 4-bit decoder). The other two (Xt1 and Xt2) are random logic circuits extracted from a commercial processor (Xtensa, from Tensilica, http://www.tensilica.com/xtensa_overview_handbook.pdf). All circuits are combinational blocks with registered POs. Synopsys' DesignCompiler performs synthesis using a 0.18-micron cell library, Cadence's Silicon Ensemble generates physical layout, Mentor Graphics' xCalibre extracts *RC* networks, and Synopsys' PrimeTime provides static timing analysis.

In each experiment, we extract the Spice netlist used in HSpice simulation from the physical layout. The netlist contains *RC* information, including coupling capacitances, so that we can observe crosstalk effects. Preliminary HSpice simulation results show that although some crosstalk effects exist at many nodes, none of the effects is strong enough to cause functional errors. In addition, we inject transient glitches of random shape and timing into the circuit to create compound noise effects with the existing crosstalk. These transient glitches can help us model various noise effects, such as an erroneous logic switch resulting from particle strikes on the transistor's sensitive region.

Because the goal is to study the vulnerability of individual circuit nodes, we focus the HSpice simulation on a specific node at a time, applying input vectors to the circuit while spuriously injecting transient glitches on a single node. These transient glitches interact with potential crosstalk noise on the node to produce compound effects and can cause observable errors, captured by the DFFs. We count the number of observable errors resulting from noise injection on each node separately, and this count serves as a measurement of the node's simulated softness. We compared these results with the ASSA-computed softness values. During computation, the actual statistics of HSpice simulation input vectors provide the input logic probability; the other two optional inputs ($F_j$ and $W_N$) are set to 1 for the lack of application-specific knowledge.

Table 1. Sample circuits and simulation time for HSpice and ASSA.

| | | Adder | DEC | Xt1 | Xt2 |
|---|---|---|---|---|---|
| Area (μm$^2$) | | 1,107 | 2,488 | 865 | 995 |
| Node count | | 89 | 210 | 74 | 59 |
| No. of simulated nodes | | 42 | 20 | 22 | 37 |
| No. of input vectors | | 512 | 128 | 512 | 256 |
| Runtime (seconds/node) | HSpice | 12,062 | 19,097 | 9,548 | 5,230 |
| | ASSA | 4.65 | 2.45 | 4.14 | 4.88 |
| Speedup factor | | 2.6 x10$^3$ | 7.8 x 10$^3$ | 2.3 x 10$^3$ | 1.1 x 10$^3$ |

Table 1 shows statistics from the experiments. Rows 1 and 2 are the area and the internal node count of the sample circuits. Rows 3 and 4 show the number of nodes on which we chose to inject noise and the number of input vectors used for simulating the effects of noise on each node. Row 5 compares ASSA's runtime with that of HSpice in terms of average evaluation time per node, and in the last row we see that ASSA achieved speedup factors on the order of 10$^3$ over HSpice. Furthermore, as circuit complexity increases, HSpice's simulation speed decreases drastically, requiring trade-offs between precision and runtime. For example, DEC is only 2.3 times the size of the adder, but the number of chosen nodes and simulated vectors had to be reduced by half and by a fourth, respectively, to finish simulation in comparable time. ASSA, however, shows constant analyzing time per node for circuits of similar complexity (adder, Xt1, and Xt2), whereas the analyzing time per node actually improves for the larger circuit, DEC.

Figure 9 compares calculated softness with simulated softness, node by node. The indices of the simulated nodes appear on the *x*-axis, and the normalized softness values of selected nodes appear on the *y*-axis, where the ASSA and HSpice curves show the ASSA-calculated and the HSpice-simulated values. As the figure

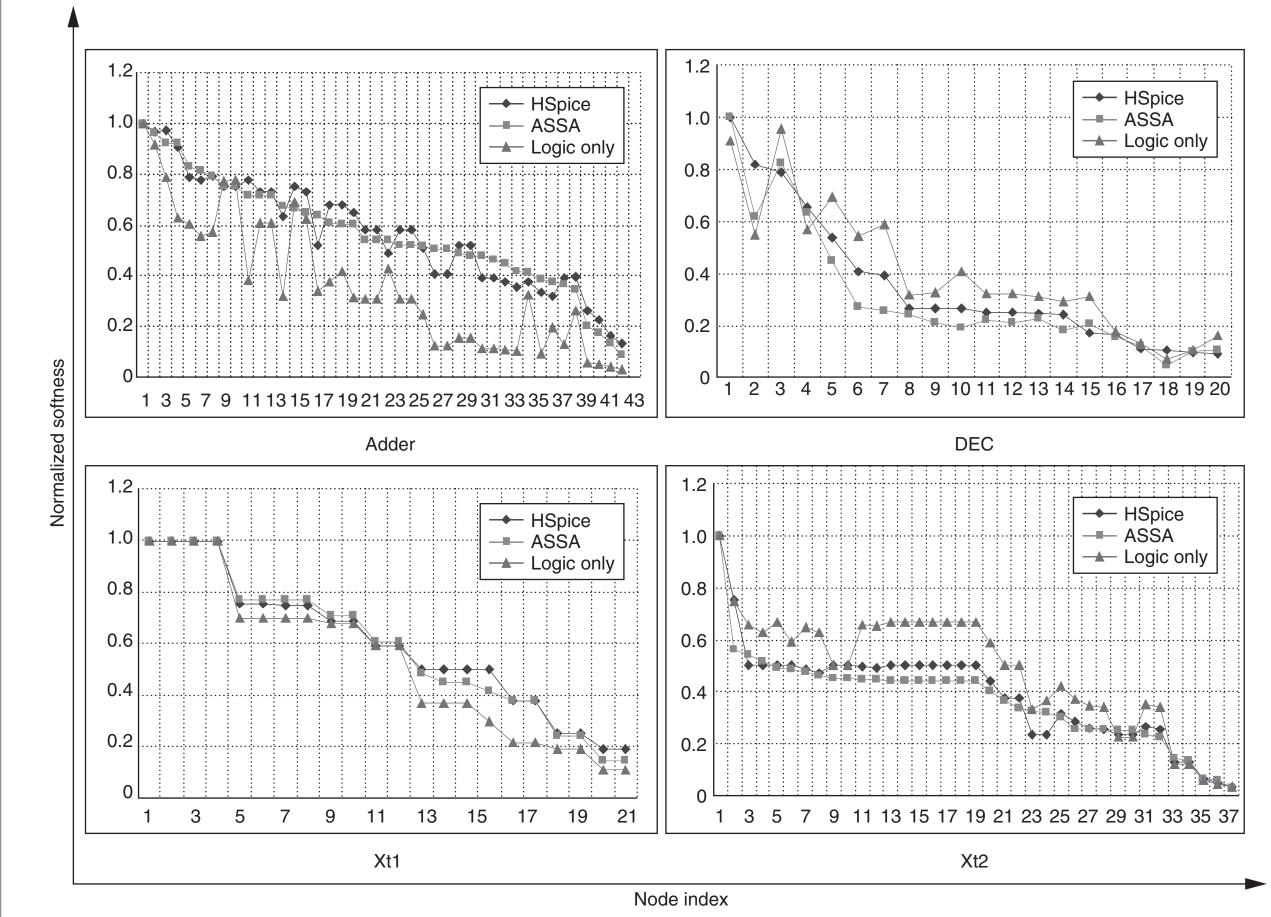


**Figure 9. Automatic soft-spot analyzer results compared with HSpice simulation.**

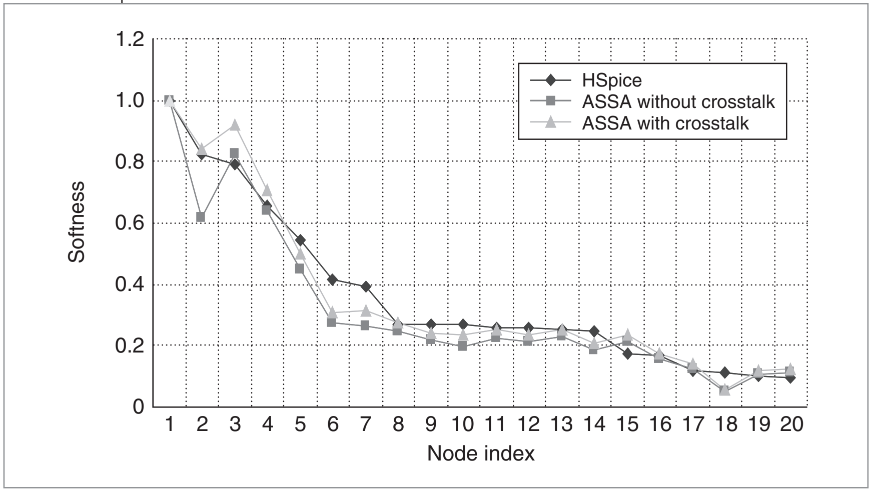


**Figure 10. Effect of considering crosstalk-induced curve shift.**

shows, ASSA not only correctly captured the most-vulnerable nodes but also provided a distribution of softness among all simulated nodes, a distribution that matched HSpice results closely. The nodes with high calculated softness values indeed cause more functional errors upon noise injection.

To demonstrate that all three masking effects contribute collectively to the overall softness evaluation and that considering only one effect is not enough, we plotted the logic masking factor in the same graph, marked "logic only." Obviously, the logic factor alone doesn't reflect the actual softness, and the correlation is highly circuit dependent. For example, for the smallest circuit, Xt1, the logic factor dominates in the overall softness. This is because all the nodes have similar capacitive loads, so their electrical factors are very close, and all timing paths have large positive slack and similar propagation delays, which means the nodes' effective noise windows are also similar. However, in a larger circuit, such as the adder, considering only the logic factor produces an inaccurate result because all three factors have large variations from node to node. The circuit's size isn't the only reason for the inaccuracy. In the largest circuit, DEC, the logic factor distribution gives a relatively higher degree of correlation than in the adder circuit. We can conclude that the logic factor doesn't consistently measure circuit softness, and the result depends heavily on the circuit's structural characteristics. The only way to measure circuit softness accurately is to jointly consider all three masking effects, using soft-spot analysis.

We obtained the ASSA calculation results in Figure 9 without considering crosstalk-induced curve shifts. As we discussed in the subsection "Electrical masking," curve shift can improve accuracy by building analyzable noise information into the NRCs. We chose the DEC circuit to demonstrate the effect of curve shift because it is relatively large and makes strong crosstalk effects observable at some internal nodes. In Figure 10, the third curve, marked "ASSA with crosstalk," represents the softness values calculated by ASSA after the analyzer considers the curve shift. The discrepancies between the simulation and the calculation results diminish as the calculated values increase for most of the nodes. At node 2 in particular, where we estimate the worst-case crosstalk to be 0.25 V, calculated softness improves from 0.62 to 0.84, which is very close to the simulated value (0.82). However, results for some nodes (such as nodes 3 and 4) indicate that considering the crosstalk effect makes the calculation more pessimistic. This is because the crosstalk estimation technique gives only the worst-case values, which might not occur during simulation.

These experiments on small circuits show that soft-spot analysis can efficiently evaluate the relative vulnerability among all nodes. However, they cannot show whether the method can handle large circuits or how the softness values are distributed among a large number of nodes. Therefore, we next apply soft-spot analysis to a large commercial circuit.

### Scalability and softness distribution

We applied ASSA to Tensilica's Xtensa, a commercial state-of-the-art configurable and extensible reduced-instruction-set computing (RISC) processor. We experimented on a large logic module, EX, with 97 registered output ports, 338 input ports, and 3,156 internal nodes. EX is particularly challenging because of its many inputs, unbalanced logic paths, and strong crosstalk effects, the latter resulting from an aggressive layout scheme. The performance and reliability requirements make it essential to identify and fix potential vulnerable spots and provide low-cost online protection schemes in the early design phase. Because of the design complexity, simulation-based methods are not applicable. Our soft-spot analysis, however, finished within a reasonable time. Table 2 shows the time breakdown for each step. The processing time for each node (2.55 seconds) is comparable to that for the DEC (2.45 seconds), indicating that our methodology can scale approximately linearly as design complexity increases.

Most of the long processing time for calculating $R_e^N$ and $TW_{\text{eff}}^N$ is due to layout, *RC* extraction, and static timing analysis. Because all these operations are inevitable steps in any VLSI design flow, our tool will perform much better and will not require much additional time or effort when integrated with a standard design flow.

We also established that the vulnerabilities of different nodes vary greatly in the large circuit EX. Figure 11a shows the softness distribution of all nodes in EX. The *x*-axis is the normalized softness in the logarithm scale (normalized to 10,000 for convenience of depiction), and the *y*-axis is the number of nodes with the various softness values. The figure clearly demonstrates the nonuniform softness distribution among all nodes: Only about 0.7% (22) have softness exceeding 10% of the maximum value, and another 18.6% (587) have softness exceeding 1%, whereas the softness values of the other 80.7% of all nodes are at least two orders of magnitude lower than the maximum value. Figure 11b shows the actual softness distribution among all nodes: The *x*-axis is the list of all circuit nodes, and the *y*-axis shows the softness values (normalized to 10,000) in the logarithm scale. This example also illustrates the selection of soft spots discussed in the subsection "Evaluating softness and identifying soft spots": If $S_{\text{th}}$ is set to 10% of maximum softness value $S_{\text{max}}$, ASSA categorizes only 22 spots as soft spots, whereas if $S_{\text{th}}$ is set to 1% of $S_{\text{max}}$, ASSA categorizes 609 nodes as soft spots.

This unbalanced softness distribution plays an important role in efficient, low-cost, robust circuit design. During the premanufacturing design phase, designers can perform accurate but time-consuming analysis on fewer soft spots. If aggressive design causes high vulnerability at some circuit nodes, localized design modifications can reduce their softness. Furthermore, our methodology provides a guideline for selectively applying an online error detection and protection scheme to the spots most likely to be affected by transient errors during the product's lifetime, so a high degree of online robustness is achievable with low design overhead.

## Applications of soft-spot analysis

We propose two useful applications of soft-spot analysis: *robustness enhancement* and *robustness insertion*. Both use the result of soft-spot analysis to efficiently improve overall circuit reliability while keeping the incurred design overhead under control, but they take different approaches.

Robustness enhancement increases a circuit's noise immunity by reducing the three masking effects at the identified soft spots through localized and limited design modifications at the gate level. As the analysis identifies soft spots, reducing one or more of the three contributing factors can reduce the spots' softness. If a

**Table 2. ASSA runtime on circuit EX.**

| Operation | Time |
|---|---|
| Calculating electrical factor $R_e^N$ | 52 minutes*+ |
| Calculating logic factor $P_{\text{prop}}^N$ | 2 minutes |
| Calculating timing factor $TW_{\text{eff}}^N$ | 78 minutes++ |
| Calculating softness $S_N$ | 2 minutes |
| Total time | 134 minutes |
| Processing time per node | 2.55 seconds |

* Library calibration time not included

+ Including layout and *RC*-extraction time

++ Including static timing analysis time

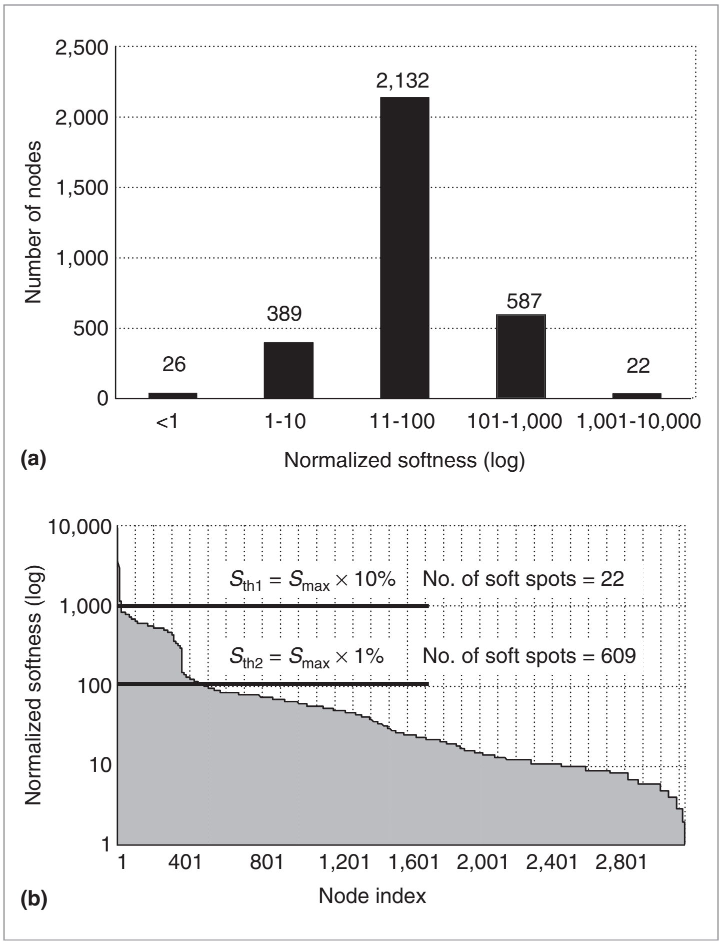


**Figure 11. Node vulnerability in the large circuit EX: softness distribution (a) and soft-spot identification (b).**

large softness value is due to a large logic factor, logic changes can reduce the propagation probability. If the timing factor is the major cause of softness, techniques such as buffer insertion can balance the timing paths and shrink the effective noise window. If the electrical factor is high, techniques such as cell resizing can improve a node's noise rejection feature. All these techniques are highly localized to the vicinity of the soft spots. However, reducing one factor might cause other factors to increase, possibly leading to a greater overall softness distribution. Therefore, researchers should develop an optimization algorithm to quickly find the best enhancement solution that optimally improves the three terms in Equation 3 with minimal circuit change.

Robustness insertion judiciously adds circuit-hardening cells at the soft spots to improve the circuit's online reliability against transient errors. Spatial and temporal redundancies that protect circuits from noise disturbances have been important techniques for improving circuit online reliability. However, without guidelines, excessive redundancy insertions incur unacceptable design overhead, and the protection might still not be efficient if the most vulnerable circuit elements are underprotected and other circuit elements are overprotected. The goal of robustness insertion is to find an optimal protection scheme to achieve the highest level of robustness improvement under given design constraints, using the guidelines of soft-spot analysis and an efficient optimization algorithm.

**AUTOMATIC SOFT-SPOT ANALYSIS** will greatly facilitate robust circuit design as circuits become more sensitive to complicated noise interferences resulting from technology scaling. It is the key first step toward the design of low-cost, highly robust nanometer circuit systems. Although the current framework has some limitations, we expect future research to make the methodology more comprehensive and more accurate.

The current methodology's applicability is limited to glitch-type noise in static digital CMOS circuits. Delay-type noise plays an equally important role in circuit reliability degradation. Unlike the glitch-type noise, which causes logic errors, delay-type noise causes timing requirement violations. Therefore, researchers must investigate ways to characterize a circuit node's softness in terms of its potential to cause excessive delays. In dynamic logic, the affecting factors differ from those in static circuits. For example, the noise rejection feature should be calibrated differently, and the timing window calculation also differs. It is imperative to extend the current framework's applicability to include a variety of circuits and noise types. ■

## Acknowledgment

This work was supported by the MARCO/DARPA Gigascale Systems Research Center.

## ■ References

1. J. Cong, D.Z. Pan, and P.V. Srinivas, "Improved Crosstalk Modeling for Noise Constrained Interconnect Optimization," *Proc. Asia and South Pacific Design Automation Conf. (*ASP-DAC 01), IEEE CS Press, 2001, pp. 373-378.
2. S. Young, "Identifying IR Drop in High Performance Nanometer Design," *Electronic Eng.*, vol. 74, no. 905, June 2002, pp. 30-33.
3. L. Shen and N. Chang, "Challenges in Power-Ground Integrity," *Proc. IEEE Int'l Conf. Computer-Aided Design* (ICCAD 01), IEEE CS Press, 2001, pp. 644-651.
4. L. Hongmei et al., "Comprehensive Frequency-Dependent Substrate Noise Analysis Using Boundary Element Methods," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 02), IEEE CS Press, 2002, pp. 2-9.
5. C. Hess et al., "Logic Characterization Vehicle to Determine Process Variation Impact on Yield and Performance of Digital Circuits," *Proc. Int'l Conf. Microelectronic Test Structures* (ICMTS 02), IEEE Press, 2002, pp. 189-196.
6. P. Hazucha and C. Svensson, "Cosmic-Ray Soft Error Rate Characterization of a Standard 0.6-μm CMOS Process," *IEEE J. Solid-State Circuits*, vol. 35, no. 10, Oct. 2000, pp. 1422-1429.
7. L. Anghel and M. Nicolaidis, "Cost Reduction and Evaluation of a Temporary Faults Detecting Technique," *Proc. Design, Automation and Test in Europe* (DATE 00), IEEE CS Press, 2000, pp. 591-598.
8. Y. Zhao and S. Dey, "Separate Dual-Transistor Registers—A Circuit Solution for On-line Testing of Transient Error in UDMS-IC," *Proc. 9th IEEE Int'l On-Line Testing Symp.* (IOLTS 03), IEEE CS Press, 2003, pp. 7-11.
9. K. Mohanram and N.A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," *Proc. IEEE Int'l Test Conf.*, IEEE CS Press, 2003, pp. 893-901.
10. L.B. Freeman, "Critical Charge Calculations for a Bipolar SRAM Array," *IBM J. Research and Development*, vol. 40, no. 1, Jan. 1996, pp. 119-129.
11. X. Bai et al., "Noise-Aware Driver Modeling for Nanometer Technology," *Proc. IEEE Int'l Symp. Quality Electronic Design* (ISQED 03), IEEE Press, 2003, pp. 177-182.

12. J.M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits—A Design Perspective*, 2nd ed., Prentice Hall, 2003.
13. T.H. Cormen et al., *Introduction to Algorithms*, McGraw-Hill, 1990.
14. S. Arora et al., "Proof Verification and the Hardness of Approximation Problems," *J. ACM*, vol. 45, no. 3, May 1998, pp. 501-555.

**Chong Zhao** is a PhD student in electrical and computer engineering at the University of California, San Diego. His research interests include signal integrity analysis and testing, circuit reliability analysis, and robust nanometer VLSI circuit design. Zhao has a BS in physics from Peking University, Beijing, China; and an MS in electrical engineering and an MA in physics from the University of Southern California, Los Angeles. He is a student member of the IEEE.

**Xiaoliang Bai** is a member of the consulting staff at Cadence Design Systems. His research interests include signal integrity analysis, circuit optimization, and design for test. Bai has a PhD in electrical and computer engineering from the University of California, San Diego. He is a member of the IEEE.

**Sujit Dey** is a professor in the Department of Electrical and Computer Engineering at the University of California, San Diego, where he heads the Mobile Embedded Systems Design and Test Laboratory. Dey is also the founder and CEO of Ortiva Wireless; he is affiliated with the California Institute of Telecommunications and Information Technology (Cal-IT2) and the UCSD Center for Wireless Communications. His research interests include developing configurable platforms consisting of adaptive wireless protocols and algorithms, and deep-submicron adaptive SoCs for next-generation wireless networks and appliances. Dey has a PhD in computer science from Duke University, Durham, North Carolina. He is a senior member of the IEEE.

■ Direct questions and comments about this article to Chong Zhao or Sujit Dey, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093, {chong, dey}@ece.ucsd.edu; or Xiaoliang Bai, baix@cadence.com.