

# A STATIC NOISE IMPACT ANALYSIS METHODOLOGY FOR EVALUATING TRANSIENT ERROR EFFECTS IN DIGITAL VLSI CIRCUITS<sup>1</sup>

Chong Zhao, Xiaoliang Bai, Sujit Dey  
{chong, xibai, dey}@ece.ucsd.edu

Department of Electrical and Computer Engineering, University of California at San Diego,  
9500 Gilman Drive, La Jolla, CA 92037. (858) 534-7883

## **Abstract**

*Single-event-upset (SEU) has become a great threat to the reliability of nanometer circuits [1]. The need for cost-effective robust circuit design mandates the development of efficient reliability analysis. In this paper, a static "Noise Impact Analysis" methodology is developed to estimate the circuit vulnerability. First, both the circuit elements and the transient noise are abstracted in the format of matrices. Then the circuit-noise interaction is modeled by a series of matrix transformations, which jointly considers three masking effects that can potentially prevent transient noise from causing observable errors. Finally, the error-resiliency of the sequential elements is considered in determining the impact of transient noise on the circuit. Experiment results demonstrate that our technique can accurately yet quickly estimate the circuit failure rate by comparing with HSPICE simulation. The proposed methodology will greatly facilitate the economic design of robust nanometer circuit.*

## **1. Introduction**

With feature size shrinking to nanometer scale, clock frequency reaching multi-GHz and supply voltage approaching sub-voltage range, reliable functioning of VLSI circuits is being greatly threatened. This is due to the fact that noise interferences are becoming more severe and complex as the same time as the noise margin of the semiconductor device is drastically reduced. Traditionally, analyzing and ensuring circuit reliability have been relying on exhaustive dynamic simulations. However, as complexity increases to having tens of millions of transistors in a single chip, simulation-based methods are no longer applicable. Therefore, static, fast yet efficient techniques are needed to analyze the overall circuit reliability.

Based on their temporal characteristics, noise effects in VLSI circuits can be categorized into either permanent or transient. In contrast to a permanent error, which will cause permanent chip malfunction, a transient error happens occasionally and lasts for only a short period of time. The physical mechanisms of the transient noise can be very complicated and hard to determine. For example, a transient error may be caused by temporary environmental conditions such as neutron and  $\alpha$  particle, power supply and interconnect noise, electromagnetic interference and electrostatic discharge [2]. In particular, Single-Event-Upset (SEU) is the most serious transient fault. The error effects may not be persistent or repeated when the noise source disappears, therefore it is impossible to predict their existence during chip design and difficult to perform on-line diagnose during the chip's operation.

Many research works have been conducted on analysis and avoidance of SEUs. [3] studied SEU caused by the cosmic ray neutrons or  $\alpha$ -particles and [4] evaluated impact of technology scaling on SEU. They were focused on the physical mechanism and the severity of the problem, rather than proposing solutions to alleviating SEU effects. [5] estimated the SEU rate from a theoretical perspective but the result is not directly applicable to evaluating the functional impact of the error rate. On-line protections and circuit hardening technologies had been developed to mitigate the transient noise effects using spatial and/or temporal redundancies [6][7]. As every protection scheme is associated with certain design overhead, blindly applying these techniques to the entire design might not be acceptable in designs that are cost-sensitive or with tight timing/area constraints. [8] proposed the idea of partial protection to apply on-line redundancies at certain circuit locations. However, it failed to develop an efficient metric to select the location of protection insertion. In [9], a new viewpoint of analyzing the circuit noise-immunity is proposed.

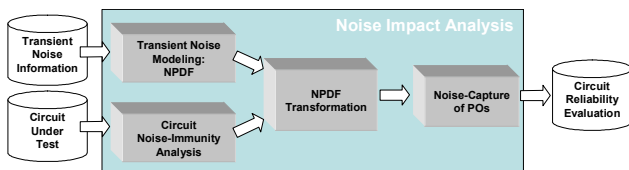
---

<sup>1</sup> This work was supported partially by the MARCO/DARPA Gigascale Systems Research Center (GSRC)

However, it only focused on the circuit characteristic but ignored the noise behavior. In summary, although extensive research work has been done, few have definitively defined a metric to numerically gauge the “reliability” of a circuit.

The reliability of a digital circuit depends on many factors that can be categorized into: (1) External factors – the behavior of the transient noise, including the probability for it to happen at different circuit location and its severity (duration and magnitude). Since transient noise is random in nature, it can be best described probabilistically. (2) Internal factors – the noise immunity of a circuit, i.e. how does a circuit respond to a transient noise and how likely will its functionality be altered accordingly. Unlike the noise behavior, which is dependent on the external environment and operating conditions, noise immunity is purely decided by the characteristics of the circuit itself and therefore it can be analyzed based on the knowledge about the design.

In order to analyze the circuit reliability, a calibration metric has to be defined and then numerically calibrated based on both the internal and external factors. In this paper, we introduce a static “Noise Impact Analysis” methodology to evaluate the reliability of static CMOS digital circuits by considering the interaction between the unpredictable external factors and analyzable internal factors. As shown in Figure 1, our methodology consists of four major parts. First, in “Transient Noise Modeling”, we introduce “Noise Probability Density Function (NPDF)” to probabilistically describe the noise occurrences. Second, in the “Circuit Noise-Immunity Analysis”, we investigate three well-known masking effects (timing, electrical and logic) and calibrate their contributions on preventing transient noise from becoming observable errors. Third, we use a “NPDF transformation” technique to study how this intrinsic noise immunity influences the distribution and propagation of the random transient noise. Finally, results from NPDF transformation are used to calculate the probabilities for the circuit primary outputs (POs) capturing transient noise. As we will discuss, this “Noise-Capture” capability is an accurate measurement of the reliability of the entire circuit.



**Figure 1. Noise Impact Analysis Framework**

Experiment results prove that our method is accurate and scalable to complex circuits. Also, we discover that the vulnerabilities of different POs to transient noise greatly vary. This variation is crucial in guiding us to find

efficient yet economical solutions to improving the circuit immunity to transient noise because previously developed on-line hardening techniques can be applied to only the most vulnerable circuit elements identified by our analysis. As a result, a high level of reliability can be achieved with limited design overhead. Furthermore, since the proposed methodology does not require dynamic vector simulation or intensive computation, it is extremely fast and capable of handling large designs.

The rest of the paper is organized as follows: section 2 introduces the transient noise modeling; section 3 discusses how to analyze the circuit noise immunity; section 4 describes the NPDF transformation and the noise-capture ratio calculation in detail; section 5 presents experimental results; and section 6 concludes the paper.

## 2. Transient Noise Modeling

A single transient noise can be modeled as a square-shaped glitch pulse  $g(w,h)$  that deviates from the correct voltage level, with width  $w$  ( $0 < w \leq T$ , where  $T$  is the clock period) and height  $h$  ( $0 < h \leq V_{dd}$ , where  $V_{dd}$  is the supply voltage) [10], regardless of the physical noise source. In this work, we will use this glitch model to describe a single transient noise that may occur due to an arbitrary transient noise source.

Transient glitches may originate at any circuit node randomly. Therefore it is desirable to describe their occurrences probabilistically. In this work, we model this randomness using a “Noise Probability Density Function (NPDF)”. For each circuit node  $N$ , we define  $NPDF(N,t,w,h)$  as the probability for a glitch  $g(w,h)$  to occur at time  $t$  ( $0 \leq t \leq T$ ). The exact form of the NPDF is related to many factors. Obtaining the noise distributions have been the target of many published works [5][10] and is beyond the scope of this work. We will assume that the initial NPDFs are known through preliminary analysis at all circuit locations prior to our analysis. However, as we will see, although the accuracy of the NPDF plays important role in the accuracy of the analysis result, the validity of our technique does not depend on the exact content of the NPDF.

Assuming a transient glitch may happen any time with equal probability, we can remove the time dependency and simplify  $NPDF(N,t,w,h)$  to  $NPDF(w,h)$  for node  $N$ . Its analytical expression might not be conveniently derived but we might numerically represent it in a table format. Figure 2 shows an example of such an NPDF table. The heights and widths of the transient glitches are labeled in the first column and the first row, respectively; the entry  $P_{ij}$  stands for the probability of the glitch having shape  $[i, j]$ . The resolution of widths and heights are set to  $1/8$  of the clock period ( $T$ ) and  $1/8$  of the power supply voltage ( $V_{dd}$ ), respectively. The unit of  $P_{ij}$  is  $1/1000$  (“‰”) and the table is normalized:

$$\sum_{[i,j] \in \text{NPDF}} P_{ij} = 1$$

For example,  $P_{35} = 10\%$  means 10 out of 1000 transient glitches at this node have the width between  $[3/8*T, 4/8*T]$  and the height between  $[5/8*V_{dd}, 6/8*V_{dd}]$ .

H \ W	0	1	2	3	4	5	6	7
0	22	16	17	15	13	17	12	18
1	21	18	15	12	19	14	15	17
2	11	14	14	13	20	20	17	19
3	11	12	16	9	18	10	16	16
4	16	19	14	13	15	8	13	16
5	18	15	14	12	18	18	20	11
6	18	13	11	14	18	16	13	14
7	13	23	21	25	16	20	14	14

Figure 2. An NPDF Example

The NPDF table is a probabilistic description of transient glitches that may originate at internal circuit nodes. However, not all glitches described in the NPDFs can cause circuit failure. Next, we will investigate the noise immunity of the digital circuit that prevents some of the transient glitches from becoming observable errors, i.e. errors that can result in circuit malfunction.

### 3. Noise-Immunity of Digital Circuits

CMOS digital circuits have intrinsic immunity to transient noise, attributed to three well-know masking effects [10]: electrical masking, timing masking and logic masking. These masking effects are strongly related to the structural features of the circuit but are independent of the occurrences of the transient noise. When propagating in a circuit, some transient noise will be killed by these effects and only transient noise that can reach the circuit POs will have erroneous impact on the rest of the system.

#### 3.1. Electrical Masking

The shape of a transient glitch will be changed as it propagates through a logic gate and noise without enough duration and amplitude will not be able to reach the output of the gate. The shape of output noise is a function of the shape of the input noise as well as the characteristics of the circuit such as the gate type, transition type and loading condition. The analytical format of the transfer function is hard to determine. Hence, we represent this function in a quantized table format called “Noise Propagation Matrix (NPM)”. Figure 3 shows an example NPM of an inverter with a load capacitance of 0.02pf for a positive-transitioned noise at its input. The rows and columns list the widths and heights of the input glitches, respectively. The entries are 2-tuples  $[W_i H_j]$  representing the width and height of the output glitch when the input noise has width  $i$  and height  $j$ . In this example, the resolution of widths and heights are set to  $1/8*T$  and  $1/8*V_{dd}$ , respectively.

H \ W	0	1	2	3	4	5	6	7
0	0,0	0,0	1,1	1,4	1,7	1,7	2,7	2,7
1	0,0	0,0	1,2	1,6	2,7	2,7	3,7	3,7
2	0,0	0,0	1,2	2,7	3,7	3,7	4,7	4,7
3	0,0	0,0	2,3	3,7	3,7	4,7	5,7	5,7
4	0,0	0,0	2,3	4,7	4,7	5,7	6,7	6,7
5	0,0	0,0	2,4	4,7	5,7	6,7	7,7	7,7
6	0,0	0,0	3,4	4,7	6,7	7,7	7,7	7,7
7	0,0	0,0	3,4	5,7	6,7	7,7	7,7	7,7

Figure 3. Example of a Noise Propagation Matrix: 0.18μm INVX1, 0 → 1, Load Capacitance = 0.02pf

Multiple NPMs are needed to completely characterize the noise response of a gate. For example, a 2-input AND gate ( $Y=A*B$ ) needs 4 NPMs for every load condition: positive and negative noise transition at A when B=1; positive and negative noise transition at B when A=1. When referring to NPMs, we will use the notation  $NPM(GATE, PIN, TRANSITION, C_{load})$ . Using this notation,  $NPM(AND, A, 0 \rightarrow 1, 0.02pf)$  refers to the NPM of at pin A of an AND gate with a loading capacitance 0.02pf facing positive-transitioned input noise. The NPMs will be used in the noise impact analysis to describe the electrical masking capability of an individual gate.

In reality, for standard-cell-based digital circuit design, NPMs of all the library cells can be pre-calibrated using SPICE simulation, stored in a database and referenced during analysis. An obvious advantage is that the calibration is done only once for a given library and can be used on all circuits mapped to the same library. Hence, we can take advantage of the accuracy of SPICE without repeatedly suffering from its time-consuming nature.

#### 3.2. Timing Masking

To become an observable error, a glitch needs to arrive at a sequential element at proper timing. Specifically, for a D-type flip-flop (DFF), it can be defined as its sampling window bounded by the setup time ( $t_{su}$ ) and the hold time ( $t_h$ ), as shown in Figure 4(a). For a circuit node  $N$ , a “noise sensitive window”  $T_N$  can be defined as such that only transient noise that occurs within this timing window may be able to reach at least one DFF within its sampling window.

As shown in Figure 4(a), the noise sensitive window of a specific timing path  $p$  has a start time ( $t_{start}^{Np}$ ) and an end time ( $t_{end}^{Np}$ ), decided by the longest ( $(d_p)_{max}$ ) and shortest ( $(d_p)_{min}$ ) propagation delay from node  $N$  to a DFF, respectively. Obviously:

$$t_{start}^{Np} = T - t_{su} - (d_p)_{max}$$

and

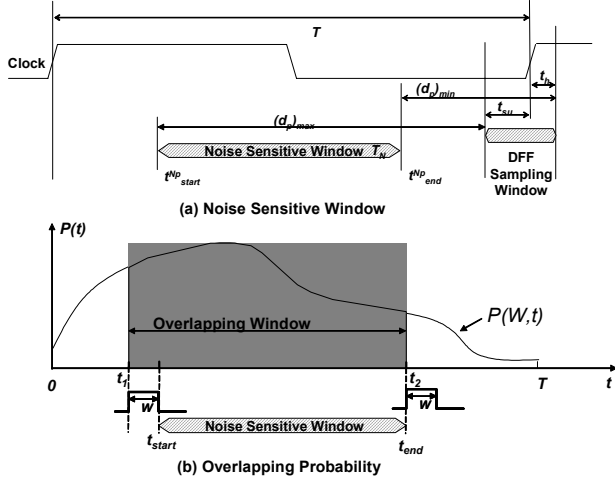
$$t_{end}^{Np} = T + t_h - (d_p)_{min}$$

The noise sensitive window is thus decided by the latest end time and the earliest start time among the collection of all timing paths  $P$ :

$$T_N = t_{end} - t_{start}$$

Where:

$$t_{start} = \min_{p \in P} \{ t_{start}^{Np} \} \quad t_{end} = \max_{p \in P} \{ t_{end}^{Np} \}$$



**Figure 4. Noise Sensitive Window**

We then determine the probability for noise at node  $N$  with width  $W$  to be sampled by DFFs, denoted as  $P_i^N(W)$ , from the probability for the noise to overlap with the noise sensitive window. As shown in Figure 4(b), for a noise to overlap with the noise sensitive window, its starting time must be earlier than  $t_{end}$  and later than  $t_{start} - W$  (if  $t_{start} < W$ , the starting time is set to be 0). During any clock period  $[0, T]$ , if the probability for noise with width  $W$  to start at time  $t$  is  $P(W, t)$ , the overlapping probability can be calculated as:

$$P_i^N(W) = \int_{t_1}^{t_2} P(W, t) dt \quad (1)$$

where  $t_1 = \max\{t_{start} - W, 0\}$ ,  $t_2 = t_{end}$ , and given the normalization requirement for noise with width  $W$ :

$$\int_0^T P(W, t) dt = 1$$

Since we assume a transient noise may happen at any time with equal probability, from the normalization condition,  $P(W, t) = 1/T$ , and:

$$P_i^N(W) = \begin{cases} T_N / T, & \text{if } W \geq t_{start} \\ (T_N + W) / T, & \text{if } W < t_{start} \end{cases} \quad (2)$$

The purpose of overlapping probability  $P_i^N(W)$  is to convert the sampling window of the destination DFFs into a local timing constraint at node  $N$ . It will be used to evaluate the timing masking effects of an individual circuit node.

### 3.3. Logic Masking

Noise at one input of a gate can not reach the output if the output logic value is determined by other inputs. The strength of the logic masking effect on an input of a gate can be measured by the probability for any other inputs carrying controlling values. To determine logic masking effect, we first calculate the probability for a circuit node  $N$  to carry 1 or 0 for each node by tracing the circuit netlist from the primary inputs (PIs) to the primary outputs (POs) and update the logic probabilities at the encountered circuit nodes during the tracing. The logic tracing technique will be discussed in more detail in the following sections. For accurate results, information on activities at the PIs should be provided, which might be derived from functional simulation vectors. Without this information, we may assume equal probability for all PIs to carry logic 1 and 0. Then a ‘‘Logic Propagation Probability’’  $P_i^N$  at the input pin of each gate, defined as the probability of no side input of the gate carrying controlling value, is calculated using the logic probabilities of all side inputs. The logic propagation probability  $P_i^N$  will be used in our analysis to evaluate the logic masking effect of an individual circuit node.

\* \* \* \* \*

In this section, we derived numerical representation of the three masking effects at individual circuit nodes:  $NPM$  for electrical masking,  $P_i^N(W)$  for timing masking, and  $P_i^N$  for logic masking. They have significant effects on how likely transient glitches can reach circuit POs with enough strength and proper timing to become observable errors, as will be discussed in the next section.

## 4. Noise Impact Analysis Using NPDF Transformation

The goal of the noise impact analysis is to estimate the impact of transient glitches on circuit functional behavior and to identify the vulnerable sequential elements that are more likely affected by these transient glitches. In large circuit systems or system-on-chips (SOCs) composed of many components, transient errors occurred inside one component will not have erroneous effects on the rest of the system unless they appear on the output ports of the component and become input errors to other components. Therefore, the reliability of this component is related to how likely the internal transient noise can cause errors at the outputs. Since the component POs are usually registered by DFFs, the problem is turned into evaluating the possibilities for these DFFs to capture noise from inside the component. We developed a static technique called ‘‘NPDF transformation’’ for this purpose. Its basic idea is to eliminate noise that can not potentially cause observable errors from the NPDF as they are being propagated in the circuit netlist and only maintain the

propagation of glitches that may be potentially captured by the PO DFFs.

Since both the transient glitches and the circuit elements are modeled as matrices (NPDFs and NPMs), transient glitches propagating through a logic gate can be viewed as NPDFs being redistributed by the NPMs while the three masking effects at the logic gate can be expressed as specific matrix operations on the NPDFs. There are three basic operations on NPDFs: (1) NPDF mapping (Section 4.1), which maps the NPDF at the input of a gate to the output using proper NPMs; (2) NPDF reshaping (Section 4.2), which realizes the noise probability redistribution due to timing and logic masking effects; and (3) NPDF superposition (Section 4.3), which combines NPDFs propagated from different locations at a single node to obtain the cumulative noise probability density at this node. The NPDF transformation (Section 4.3) consists of repetitive usage of these operations when propagating the NPDFs from the PIs to the POs.

The NPDFs obtained at the POs are used to determine how likely the logic values at each PO can be erroneously changed by the internal noise (Section 4.4). And as we will see in Section 4.5, the proposed method is highly efficient so it is applicable to very large circuit systems.

#### 4.1. NPDF Mapping

The shape of a transient glitch will change as it propagates from the input to the output of a logic gate. Once the shape of the input glitch is known, the shape of the output glitch is determined by the NPM of the gate: as shown in Figure 5, the NPDF at the output (*ONPDF*) can be viewed as the NPDF at the input (*INPDF*) being redistributed by a certain NPM (depending on the gate type, transition and load capacitance). This redistribution procedure is defined as “NPDF mapping”, as described in Figure 6: each  $INPDF[i,j]$  is mapped to a certain location  $[k,l]$  in the *ONPDF*, where the  $[k,l]=NPM[i,j]$ . When multiple locations in *INPDF* are mapped to the same location in *ONPDF*, the values accumulate, indicating a multi-to-1 mapping from the *INPDF* to the *ONPDF*. The function of NPM is analogous to an optical lens that maps an object (*INPDF*) to an image (*ONPDF*) of a different shape. The NPDF mapping procedure is used to describe the electrical masking effect.

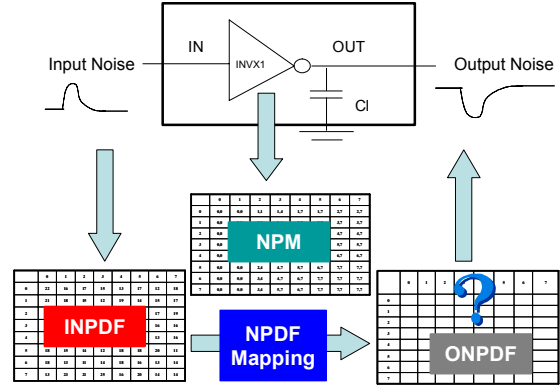


Figure 5. NPDF Mapping – Model Abstraction

```

NPDF Mapping (INPDF, NPM)
N ← The number of rows in the tables
M ← The number of columns in the tables.
/* Initialization of the ONPDF table */
for i = 0 to N-1
  for j = 0 to M-1
    ONPDF[i,j] ← 0

/* NPDF Mapping */
for i = 0 to N-1
  for j = 0 to M-1
    P ← INPDF[i,j]
    [k,l] ← NPM[i,j]
    /* If multiple location in INPDF is mapped to the same location, the result accumulates */
    ONPDF[k,l] ← ONPDF[k,l] + P
  
```

Figure 6. NPDF Mapping Procedure

Figure 7 shows the mapping of the NPDF in Figure 2 using the NPM in Figure 3. The *INPDF* and the *NPM* are re-drawn in (a) and (b), respectively. For example,  $INPDF[2,4]=20$  and  $NPM[2,4]=[3,7]$ , so  $ONPDF[3,7]$  is set to “20” (c). After mapping every  $INPDF[i,j]$ , the final *ONPDF* is shown in (d).

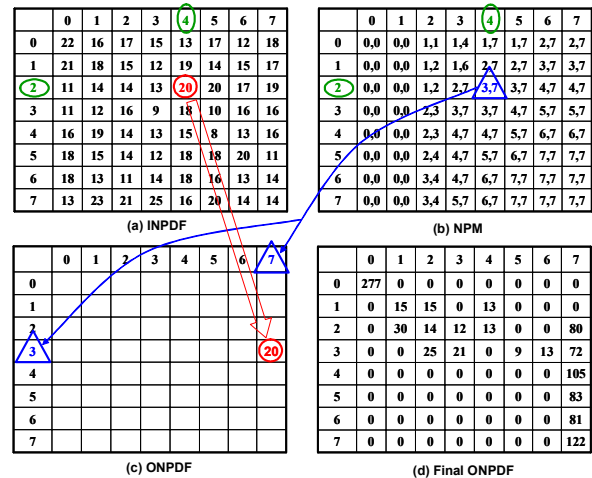


Figure 7. NPDF Mapping Example

#### 4.2. NPDF Reshaping

While the electrical masking is described by the NPDF mapping process, the effects of timing and logic factors can be best described by “NPDF reshaping”: given an NPDF and a reshaping matrix  $R$  ( $0 \leq R[i,j] \leq 1$  for all  $[i,j]$ )

of the same size, reshaping operation  $NPDF' = NPDF ** R$  is defined as:

$$NPDF'[i,j] = \begin{cases} R[i,j] * NPDF[i,j], & \text{if } [i,j] \neq [0,0] \\ NPDF[0,0] + \sum_{[k,l] \neq [0,0]} (1 - R[k,l]) * NPDF[k,l] & \text{if } [i,j] = [0,0] \end{cases} \quad (3)$$

The reshaping operation scales every  $NPDF[i,j]$  by a fraction ( $R[i,j]$ ) and moves the rest ( $1-R[i,j]$ ) to  $NPDF[0,0]$ . For example, if the NPDF shown in Figure 7(d) is to be reshaped by a reshaping matrix with all  $R[i,j]$ 's equal to 20%, then values in all locations except  $[0,0]$  is multiplied by 20% and the rest 80% is accumulated to  $[0,0]$ . The effects of logic and timing masking on noise propagation can be realized by reshaping an NPDF with proper reshaping matrices.

The logic reshaping matrix is defined as  $R_i^N[i,j] = P_i^N$  for all  $i$ 's and  $j$ 's, where  $P_i^N$  is the logic propagation probability obtained in Section 3.3. It means noise at the input of a gate can logically reach the output only when all the other inputs have non-controlling values, whereas noise with all shapes will cease to propagate if any of the other inputs carries controlling value so they should be eliminated from further propagation. Reflected in the NPDF operation, it is equivalent to relocate this portion of noise to location  $[0,0]$ .

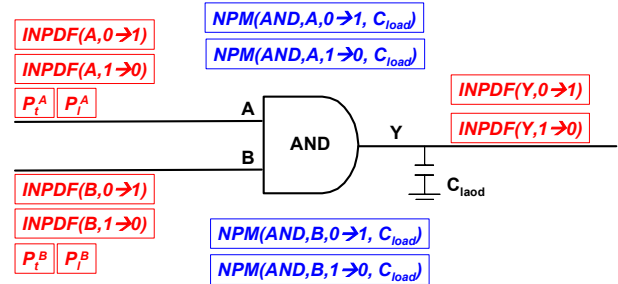
The timing reshaping matrix  $R_i^N$  is defined as:  $R_i^N[i,j] = P_i^N(i)$  for all  $i$ 's and  $j$ 's, where  $P_i^N(i)$  is the overlapping probability defined in equation (2). Unlike logic reshaping, the content of the timing reshaping matrix  $R_i^N$  is different for different rows because the overlapping window is a function of the input noise width. The timing reshaping operation on NPDFs means only a portion ( $P_i^N(i)$ ) of the noise with width  $i$  can reach a DFF within its sampling window so the rest should be eliminated from further propagation, which is equivalent to relocate the reset of the noise to location  $[0,0]$ .

### 4.3. NPDF Transformation

The NPDF transformation consists of a series of mapping and reshaping operations at individual gates as well as superposition of NPDFs propagated from different locations. To describe NPDF transformation, we use a 2-input *AND* gate (Figure 8(a)) as an example. The input pins are *A* and *B*, the output pin is *Y*, and the load capacitance is  $C_{load}$ . We assume each circuit node has well-defined initial NPDFs.

In the first step, INPDFs at all circuit nodes (*A*, *B* and *Y*) are timing-reshaped. This is to eliminate noise originated at all nodes that will not be able to reach any DFF within the sampling window. For example, the positive-transition NPDF at input *A* is reshaped by  $R_i^A$ :  $INPDF_i(A,0 \rightarrow 1) = INPDF(A,0 \rightarrow 1) ** R_i^A$ .

In the second step, the  $NPDF_i$ 's at *A* and *B* are logic-reshaped. This is to eliminate noise at inputs that will be logic-masked from propagation. For example,  $INPDF_i(A, Y, 0 \rightarrow 1)$  is reshaped by  $R_i^A$ :  $NPDF_i(A, Y, 0 \rightarrow 1) = INPDF_i(A, Y, 0 \rightarrow 1) ** R_i^A$ , where  $R_i^A[i,j]$  equals to the probability the side input *B* having non-controlling values ("1" for an AND gate). Logic reshaping of NPDF at *Y* is not done in this step because its logic reshaping matrix can not be determined until we have knowledge of the gate(s) it drives.



(a) NPDF Transformation: AND Gate Example

```

/* NPDF transformation of the positive-transition NPDF from input pin A to output pin Y of a 2-input
AND gate*/
NPDF_transformation(INPDF(A,0→1), P_i^A, P_i^A, NPM(AND,A,0→1, C_load))
begin
  N ← Number of rows
  M ← Number of columns
  /* Step 1. Re-shaping the input NPDF by the timing reshaping R_i^A */
  for i = 0 to N-1
    for j = 0 to M-1
      R_i^A[i,j] = P_i^A(i) /* Construct the re-shaping matrix from the overlapping probability. */
  INPDF(A,0→1) = INPDF(A,0→1) ** R_i^A
  /* Step 2. Re-shaping the result NPDF by the logic reshaping matrix R_i^A */
  for i = 0 to N-1
    for j = 0 to M-1
      R_i^A[i,j] = P_i^A /* Construct the re-shaping matrix from the logic propagation probability */
  INPDF(A,Y,0→1) = INPDF(A,0→1) ** R_i^A
  /* Step 3. Mapping the reshaped INPDF to the output using the proper NPM */
  ONPDF(A,Y,0→1) =
    NPDF_Mapping(INPDF(A,0→1), NPM(AND,A,0→1, C_load))
end

```

(b) NPDF Transformation: Procedure Pseudo-code

Figure 8. NPDF Transformation Example

In the third step, the  $INPDF_i$ 's at *A* and *B* are mapped to output *Y* using the  $NPDF\_Mapping$  procedure defined in Figure 6. For example,  $INPDF_i(A,0 \rightarrow 1)$  is mapped to  $ONPDF(A, Y, 0 \rightarrow 1)$  via  $NPM(AND, A, 0 \rightarrow 1, C_{load})$ . This step is to determine how the distribution of noise (that "survives" the timing and logic masking effects) will be changed due to the electrical masking effect.

Figure 8(b) shows the procedure of transforming an INPDF at input *A* to output *Y* for a positive glitch. Since logic gates usually have different response to positive and negative transition noise (as characterized by different NPMs), transformations of both type transitions need to be done. As a result, four ONPDFs are generated at output *Y*. Finally, ONPDFs from pin *A* and *B* of the same transition types are merged with INPDF of the same

transition type at  $Y$  by the “NPDF superposition” to obtain the “cumulative NPDF”:

$$C\_NPDF(Y, 0 \rightarrow 1)[i, j] = INPDF_i(Y, 0 \rightarrow 1)[i, j] + ONPDF(A, Y, 0 \rightarrow 1)[i, j] + ONPDF(B, Y, 0 \rightarrow 1)[i, j] \quad (4)$$

$$C\_NPDF(Y, 1 \rightarrow 0)[i, j] = INPDF_i(Y, 1 \rightarrow 0)[i, j] + ONPDF(A, Y, 1 \rightarrow 0)[i, j] + ONPDF(B, Y, 1 \rightarrow 0)[i, j] \quad (5)$$

Note that the NPDF at  $Y$  used in the superposition is the timing-reshaped  $NPDF_i$ . This is because the timing masking is local to a certain node once the overlapping window at the node is obtained, so its effect can be considered to the individual node prior to the propagation. In contrast, the logic masking effect of a gate applies to all the glitches that arrive at its input so the logic reshaping has to be performed as the NPDFs propagate. For example, the cumulative NPDF at  $Y$  calculated in equation (4) and (5) will be logic-reshaped when it propagates to the input of its driving gate(s).

The  $C\_NPDFs$  contain information about the distribution of noise at node  $Y$ . It includes not only noise that may be originated at  $Y$ , but also noise propagated from  $A$  and  $B$ . Note that for a negating gate, the polarity of the ONPDFs propagated from the inputs needs to be inverted before merging with the INPDF at the output to obtain the cumulative NPDF.

For a circuit consisting of multiple gates and many levels of logics, the logic-reshaping, mapping and superposition operations need to be performed repetitively as the NPDF travels in the circuit netlist from PIs to POs. In order to achieve optimal processing speed, an efficient netlist tracing algorithm is developed based on breadth-first search (BFS) [10]: The circuit is first converted to a directed graph where the logic gates are represented by vertices and wires by the edges going from the driving gate to the driven gate(s). Then starting from PIs (roots), the frontier is expanded between discovered and undiscovered vertices uniformly across the breadth of the frontier. Transformations of NPDFs from the predecessors are performed at each vertex on the frontier. This process is repeated until the BFS reaches all POs (leaves).

#### 4.4. Evaluating the “Noise-Capture Ratio” of POs

A complex digital VLSI circuit system may consist of many components. The primary outputs of each component are usually registered using DFFs to ensure glitch-free output signals. Due to their inherent noise-resistant capabilities, DFFs are only sensitive to noise with enough width and height. It means only noise described in certain region of the NPDFs propagating to these POs will be captured. This region is defined as the “dangerous zone” as show in Figure 9: only noise with shape inside the dangerous zone can be captured by the destination DFF. The dangerous zone is an inherent

characteristic of the DFF cell and is independent of the values in the NPDF, therefore it can be calibrated prior to performing noise analysis.

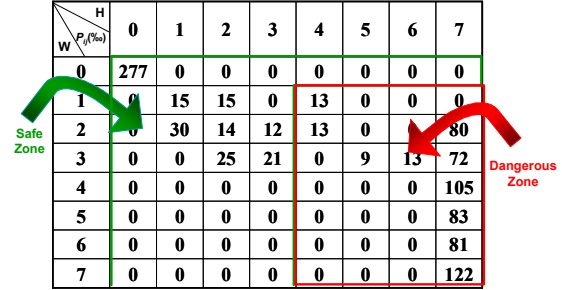


Figure 9. DFF Dangerous Zone in NPDF

Applying the dangerous zone of a DFF to the NPDF that arrives at its input pin, we are able to measure its “noise-capture ratio”  $R_c$ :

$$R_c = P^D(0) * \sum_{[i,j] \in DZ} NPDF_{pos}[i,j] + P^D(1) * \sum_{[i,j] \in DZ} NPDF_{neg}[i,j] \quad (6)$$

where  $P^D(0)$  and  $P^D(1)$  are the probabilities for logic 0 and 1 to appear at the DFF input pin  $D$ , respectively; the summations are taken over all entries in the dangerous zone (“DZ”) in the positive- and negative-transition NPDFs. A weighted sum of the two different polarities is needed because a positive glitch can happen only if the nominal value is 0, and a negative glitch can happen only if the nominal value is 1.

The DFF noise-capture ratio of each circuit PO is the primary result of the noise impact analysis. It provides an important and explicit metric to gauge relative robustness of different POs. A larger noise-capture ratio indicates a higher vulnerability to transient glitches.

#### 4.5. The Efficiency of Noise Impact Analysis

In this subsection, we discuss the efficiency of NPDF transformation technique by considering the efforts required in preparing the necessary inputs and the execution efficiency.

Prior to NPDF transformation, some information about the circuit needs to be prepared: NPM calibration is done only once for a cell library. Load capacitances on all wires are extracted from physical layout; and path timings are derived from static timing analysis (STA). Both steps are part of any standard digital design flow so no additional effort is needed. Hence, the noise impact analysis requires minimal extra effort of input preparation.

During NPDF transformation, the operations at an individual circuit node are simple (table lookup, basic arithmetic, etc.) and the processing time is constant. Therefore an efficient logic tracing algorithm is the deciding factor in the efficiency of the technique. The complexity of our BFS-based logic tracing algorithm is

$O(V+E)$ , where  $V$  is the number of vertices (gates) and  $E$  is the number of edges (wires) in the directed graph. Since our technique only requires tracing the circuit netlist once, the high efficiency is obvious because a systematic trace of the netlist is the minimum requirement for any circuit analysis.

## 5. Experimental Results

We have done several experiments to verify the accuracy, efficiency and scalability of the proposed methodology. In the first two experiments, results from the noise impact analysis are compared with HSPICE simulation. In the third experiment, the methodology is applied to a large circuit to demonstrate its efficiency and scalability.

All experiments are done on a 2.5GHz Pentium4 processor with 512MB RAM running Linux Redhat7.0. The circuit-under-tests (CUTs) are synthesized to a  $0.18\mu\text{m}$  standard cell library using Synopsys DesignCompiler<sup>TM</sup>; Synopsys PrimeTime<sup>TM</sup> is used for static timing analysis; Cadence SiliconEnsemble<sup>TM</sup> is used for physical layout; Mentor Graphics Xcalibre<sup>TM</sup> is used for RC extraction – it is needed to determine the loading capacitances on all circuit nodes in order to retrieve the proper NPMs from the pre-calibrated NPM database. The NPDF transformation is implemented in C++ and the entire flow is linked by TCL scripting language.

### 5.1. Experiment I: One Simple Circuit

The first experiment is intended to validate the NPDF transformation technique by running HSPICE simulation on a very simple circuit (Figure 10). During the simulation, each of all eight input combinations (“000” – “111”) was applied to the circuit for 5000 consecutive clock cycles, and during each cycle, a glitch with random shape and timing was injected on one of the five nodes (IN1, IN2, IN3, E and F), so each node was disturbed 1000 times for each input vector. The glitches measured at node OUT were categorized according to their shapes to generate the simulated cumulative NPDFs. It was then compared with the cumulative NPDF calculated by NPDF transformation. The INPDFs at all internal nodes used in the NPDF transformation were generated based on the distributions of the shapes of the glitches injected during HSPICE simulation.

Figure 11 shows the simulated and calculated NPDFs at node OUT. The DFF noise-capture ratio  $R_c$  calculated using equation (6) are also listed under the tables for comparison. The DFF dangerous zones are outlined in the NPDFs as well (notice that the DFF has different dangerous zones for positive and negative noise). It can be seen that both matrices produced by NPDF transformation match the simulation result very well and the calculated and simulated  $R_c$  values are within 7% of

each other. The difference between the exact values is partly caused by the loss of precision when the tables are quantized. Increasing the table resolution will result in more accurate result, but at the cost of more memory.

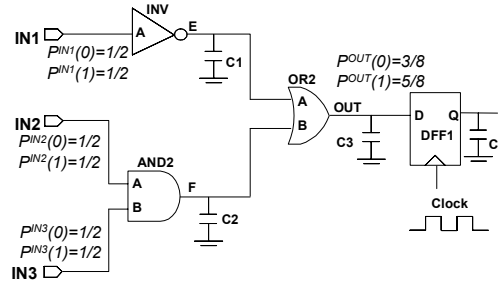


Figure 10. Simple Circuit used in Experiment I.

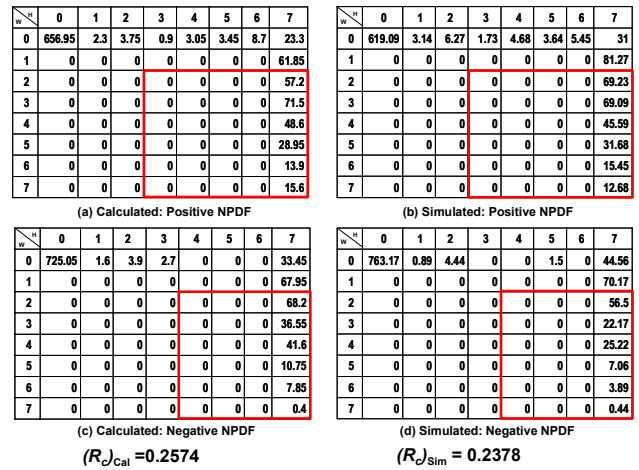


Figure 11. Experiment I Results

### 5.2. Experiment II: Two Larger Circuits

The second experiment is to validate the accuracy of the DFF noise-capture ratio using two larger circuits: CUT1 has 4 PIs, 7 internal nodes, 15 combinational gates and 8 DFF-registered POs. CUT2 has 4 PIs, 12 nodes, 21 gates and 8 POs. In both cases, simulations were run with all 16 input combinations (“0000” – “1111”). For each input vector, 1000 glitches with random shapes were injected on each internal node. The number of errors captured by each DFF was recorded to obtain the simulated noise-capture ratio: if the total number of glitches injected is  $M$  and  $m_i$  errors are detected in the  $i^{th}$  DFF, then  $(R_c^i)_{sim} = m_i/M$ . The noise impact analysis was performed on both circuits using the same noise distribution used in the simulation. NPDFs arriving at the input of each DFF were obtained from NPDF transformation technique;  $(R_c^i)_{cal}$  was calculated using equation (6).

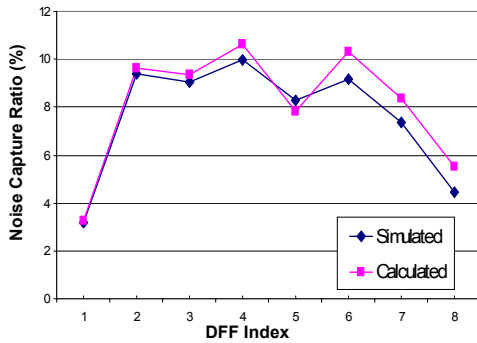
Table 1 shows the experimental results on CUT1: “DZ(pos)” and “DZ(neg)” are the sum of entries in the dangerous zones in the positive and negative transition NPDF at the DFF inputs, respectively; next column lists the logic probabilities at the DFF inputs; the calculated

and simulated  $R_c$ 's are listed in the next two columns; and the last column lists the relative error between the calculated and simulated results: the discrepancy is within 10% for most of the POs.

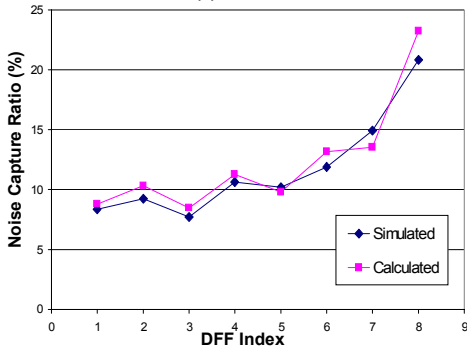
Figure 12 illustrates the calculated and simulated noise-capture ratio for both circuits: the x-axis is the indices of the 8 DFFs; the y-axis is the DFF noise-capture ratio values. A high degree of correlation can be observed in the comparison.

**Table 1 CUT1:  $R_c$  Comparison with HSPICE**

PO	DZ(pos)	DZ(neg)	Pr(1)	Cal. $R_c$	Sim. $R_c$	Err
1	0.0480	0.0276	0.75	0.0327	0.0317	3.0%
2	0.1242	0.0744	0.5625	0.0962	0.0940	2.3%
3	0.0983	0.0905	0.625	0.0934	0.0906	3.0%
4	0.1222	0.0938	0.5625	0.1062	0.0997	6.2%
5	0.0749	0.0817	0.5	0.0783	0.0828	5.8%
6	0.1296	0.0594	0.375	0.1033	0.0915	11%
7	0.0945	0.0510	0.25	0.0836	0.0735	12%
8	0.0587	0.0311	0.125	0.0552	0.0501	9.3%



(a) CUT1



(b) CUT2

**Figure 12.  $R_c$  Comparison with HSPICE**

Due to memory limitation, HSPICE simulation had to be partitioned into multiple runs. It took a total of ~5480 minutes to finish the entire simulation on CUT1 and ~6824 minutes for CUT2. In contrast, given all required input, the NPDF transformation takes only ~2 seconds for

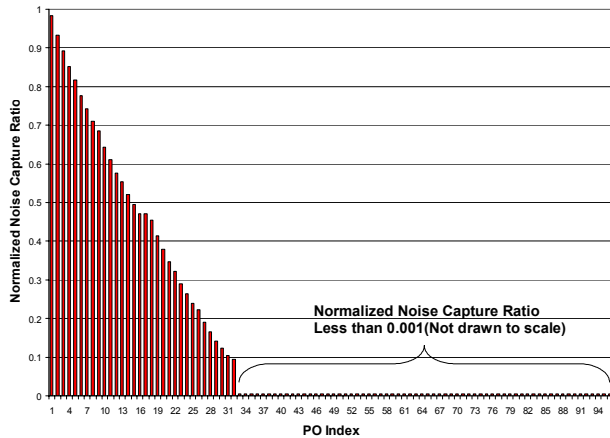
CUT1 and ~3 seconds for CUT2. A  $10^5$  speed-up is achieved over the simulation-based approach.

As we mentioned in Section 2, we assume the initial NPDFs are available to us prior to the noise impact analysis. Obtaining accurate initial NPDFs requires additional analysis which is not part of our methodology. It is worth mentioning that the validity of our methodology does not depend on the input NPDFs because it always measures the PO vulnerabilities under given noise condition. The PO with highest noise-capture ratio under certain noise condition might no longer be the most vulnerable one under different noise condition. As in our experiments, without any specific knowledge of noise condition, the noise impact analysis results accurately measures the circuit PO's vulnerability under a pure random noise distribution. In reality, the more accurate the input NPDFs that describe the noise condition, the more accurate the noise impact analysis can measure the circuit vulnerability.

### 5.3. Experiment III: Scalability to Large Circuits

We could not use larger circuits for comparison simply because HSPICE simulation can no longer be performed. However, the proposed methodology is a scalable approach that can be applied to large complex designs. To demonstrate this, we applied it to the Xtensa™ processor [12], a commercial state-of-the-art configurable and extensible RISC processor. Experiment is conducted on a large logic module EX, which has 97 registered output ports, 338 input ports and 3156 internal nodes. Given all necessary design information, the NPDF transformation calculated  $R_c$  for all 97 POs within ~16 seconds. It proves that our technique is of extremely high efficiency. In fact, as the circuit size becomes larger, we expect even better performance in terms of analyzing time per node because the time for some program setup, such as reading in the NPM database, is independent of the design size.

We discovered that the noise-capture ratio of different POs vary greatly in a large circuit. As shown in Figure 13, where the noise-capture ratio (normalized to the largest value) is plotted for all POs, the highest  $R_c$  is 3 orders of magnitude higher than the lowest value and only a small percentage of all the POs have very high  $R_c$  value. About 68% of all POs have negligible noise-capture ratio, which means the possibilities for these POs to capture transient noise from the internal circuit nodes are significantly lower than the others (The negligible  $R_c$ 's in the figure are not drawn to scale and are enlarged by 10x, simply for the purpose of making them visible on the graph).



**Figure 13.  $R_c$  Distribution in EX**

This unbalanced distribution of PO vulnerability means that we can use the noise impact analysis result to guide cost-effective reliable circuit design. In reality, it might not be acceptable to protect all circuit POs due to factors such as design cost, area/timing overhead. With the guidance of our noise impact analysis, circuit-hardening techniques can be applied to the most vulnerable POs with high noise-capture ratios. Although this will not make a circuit 100% immune to transient noise, it can achieve optimal protection result under given design budget [13].

## 6. Conclusion

In this paper, a novel noise impact analysis methodology based on an efficient NPDF transformation technique is developed to evaluate the circuit vulnerability to transient noise. It provides a promising solution to effectively analyzing circuit robustness, which leads to economically designing highly reliable digital systems in a guided manner.

## 7. References

[1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors, 2003.

[2] Jeong-Taek Kong, "CAD for Nanometer Silicon Design Challenges and Success," *IEEE Trans. VLSI Systems*, pp. 1132-1147, Vol. 12, No. 11, Nov. 2004.

[3] Peter Hazucha, Christer Svensson, "Cosmic-Ray Soft Error Rate Characterization of a Standard 0.6- $\mu\text{m}$  CMOS Process," *IEEE Jnl. Solid-State Circuits*, vol. 35, no. 10, Oct. 2000.

[4] R.C. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *Digest of International Electron Devices Meeting*, pp. 329-332, 2002.

[5] Ming Zhang, Naresh R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," *ICCAD'04*, San Jose, CA, Nov. 2004.

[6] Anghel, L., Nicolaidis, M., "Cost Reduction and Evaluation of a Temporary Faults Detecting Technique," *DATE '00*, pp. 591-598, March 2000.

[7] Y.Zhao, S.Dey, "Separate Dual Transistor Register-an Circuit Solution for on-line Testing of Transient Errors in UDSM-IC," *IOLTS'03*, pp.7-11, Kos Island, Greece, June 2003.

[8] K. Mohanram, N.A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," *IEEE International Test Conference*, pp. 893-901, Sept., 2003.

[9] C.Zhao, X. Bai, S.Dey, "A scalable soft spot analysis methodology for compound noise effects in nanometer circuits," in *Proc. of 41<sup>st</sup> Design Automation Conference*, pp. 894-899, San Diego, CA, June, 2004.

[10] P. Shivakumar and et al., "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Int. Conf. Dependable Systems and Networks*, 2002, pp. 389-398.

[11] Ch 23, T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms", McGraw-Hill, 1990.

[12] [http://www.tensilica.com/xtensa\\_overview\\_handbook\\_k.pdf](http://www.tensilica.com/xtensa_overview_handbook_k.pdf), *Xtensa<sup>TM</sup> Microprocessor Overview Handbook*, Tensilica Inc, August 2001.

[13] Chong Zhao, Yi Zhao, Sujit Dey, "Constraint-Aware Robustness Insertion for Optimal Noise-Tolerance Enhancement in VLSI Circuits," in *Proc. of 42<sup>nd</sup> Design Automation Conference*, pp.190-195, Anaheim, CA, June 2005.