Independent and Interdependent Latch Setup/Hold Time Characterization via Newton–Raphson Solution and Euler Curve Tracking of State-Transition Equations

Shweta Srivastava and Jaijeet Roychowdhury

***Abstract*—Characterizing setup/hold times of latches and registers, which is a task crucial for achieving timing closure of large digital designs, typically occupies months of computation in semiconductor industries. We present a novel approach to speed up latch characterization by formulating the setup/hold time problem as a scalar nonlinear equation $h(\tau_s, \tau_h) = 0$; this nonlinear algebraic formulation is derived from, and embeds within it, the state-transition function of the latch. We first present a technique to characterize setup and hold times independently of each other: by decoupling $h(\tau_s, \tau_h) = 0$ into two equations $h(\tau_s) = 0$ and $h(\tau_h) = 0$ and solving each equation using the Newton–Raphson method. Next, we also present a method for *interdependent* characterization of latch setup/hold times—a core component of techniques for pessimism reduction in timing analysis. We achieve this by solving the *underdetermined* nonlinear equation $h(\tau_s, \tau_h) = 0$ using a Moore–Penrose pseudoinverse-based Newton method. Furthermore, we use null-space information from the Newton's Jacobian matrix to efficiently find constant-clock-to-q contours (in the setup/hold time plane) via an Euler–Newton curve-tracing procedure. We validate fast convergence and computational advantage for independent characterization on transmission gate and C^{2}MOS latch/register structures, obtaining speedups of $2.5-10\times$, at high levels of accuracy, over the current standard of binary search. We validate the method for interdependent characterization on true single-phased clock and C^{2}MOS, obtaining speedups of more than $10\times$ for tracing 17–24 points, over prior approaches while achieving superior accuracy; this speedup linearly increases with the precision with which curve tracing is desired. We also apply our method for interdependent characterization on a transmission gate register to illustrate limitations of our method.**

***Index Terms*—Characterization, Euler–Newton, hold time, interdependence, Newton–Raphson, setup time.**

Manuscript received March 3, 2007; revised September 7, 2007. This work was supported in part by the Strategic Computer-Aided Design Laboratories, Intel Corporation, by MARCO/GSRC, by the National Science Foundation, and by the Semiconductor Research Corporation. This paper was recommended by Associate Editor S. Nowick.

S. Srivastava was with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455-0213 USA. She is now with Synopsys, Inc., Mountain View, CA 94043-4024 USA (e-mail: shwetas@umn.edu).

J. Roychowdhury is with the Department of Electrical and Computer Engineering and the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455-0213 USA (e-mail: jr@umn.edu).

Digital Object Identifier 10.1109/TCAD.2008.917595

I. Introduction

FINDING the setup and hold times of latches and registers is a crucially important prerequisite for static and dynamic timing analysis of digital circuits [1]–[3]. As devices shrink, clock speeds become ever faster, and design margins become increasingly squeezed in high-speed digital systems, it becomes important to determine these quantities with high accuracy to ensure that timing analysis makes neither unduly optimistic nor pessimistic predictions [1], [4]. Optimism in setup/hold times can cause failure in fabricated circuits, whereas pessimism leads to inferior performance. As a result, full SPICE-level transient analysis of latch/register circuits, using the best and most detailed device models available, has been emerging as the only reliable means of timing characterization for cutting-edge industrial designs.

The computational expense of these SPICE-level simulations in current industrial practice is extremely high. Setup/hold times need to be characterized for every register/cell of every standard cell library, each typically containing hundreds or thousands of cells, for all process–voltage–temperature (PVT) corners or statistical process samples. Characterization typically takes weeks or months even on large dedicated computer clusters. Therefore, even relatively modest improvements in core characterization procedures can have a large impact on reducing the time taken to achieve timing closure and on the quality of designs achieved.

In general practice today, setup and hold times are independently determined, i.e., it is assumed that the two quantities are not correlated with each other. However, the fact that setup and hold times are interdependent is, in fact, well known (e.g., [1]); that is, multiple pairs of setup and hold times are possible, which result in the same clock-to-q delay.[1] Flexibility in trading off setup versus hold time is important for reducing violations in static timing analysis [1] without sacrificing performance. As will be described in more detail later in this paper, finding interdependent pairs of setup/hold times via the new characterization method presented here results in speed improvements of some $10\times$ over prior approaches. More importantly, the technique can be used to find independent setup/hold times, with speedups of $2.5-10\times$, depending on the accuracy required. In

[1]See Section II for an explanation of setup/hold times, clock-to-q delay, and other relevant concepts.

Sections I-A and -B, we review the independent and interdependent setup/hold problems, outline our differential contributions, and summarize the key points and features of our approach.

A. Independent Setup/Hold Times

The prevalent technique for finding independent setup/hold times is to find clock-to-q delays for various trial setup/hold skews via a series of transient simulations embedded in a binary search process. In contrast, in this paper, we adapt ideas from mixed-signal/RF simulation [5]–[7] to propose a new technique for finding setup/hold times, by expressing them as the solution to a scalar nonlinear equation $h(\tau) = 0$, where τ can be the setup or the hold time. Our formulation uses the nonlinear state-transition function [8], [9] of the differential equations describing the latch and incorporates a threshold condition to detect onset of metastability.

We numerically solve the equation $h(\tau) = 0$ using the well-known Newton–Raphson (NR) method [10]. Because NR has the property of quadratic convergence as it approaches the solution, it is able to "zoom in" on the correct solution (i.e., increase accuracy) much more rapidly than binary search. As a result, it provides significant computational advantage over binary search in higher accuracy regimes, as we demonstrate in Section V. Computational advantage can result in spite of the fact that NR on $h(\tau) = 0$ requires computation of the derivative $dh/d\tau$—indeed, this extra gradient information is a core differentiator against binary search and is crucially responsible for NR's convergence. (Readers familiar with RF/mixed-signal simulation will note connections with shooting methods [5]–[7] and with transient sensitivity computation [11], [12] in Section IV and Appendix A, which contains mathematical and algorithmic details of the technique.)

We validate the new method in detail using two prototypical register designs: 1) a transmission-gate-based master–slave register and 2) a C^{2}MOS edge-triggered register. Our experiments confirm that NR's gradient-directed quadratic convergence results in speed advantages (of $\sim 2.5-10\times$), particularly at higher accuracies.

B. Interdependent Setup/Hold Times

The currently prevalent method [1] for finding interdependent pairs of setup/hold times is to first obtain (using many transient simulations of the latch) the clock-to-q delays corresponding to many trial combinations of setup and hold skews, i.e., a clock-to-q *delay surface*. This is followed by extraction of a contour in the setup/hold time plane that contains all points that result in a prescribed increase (e.g., 10% is typical) in the clock-to-q delay. Alternatively, interdependent pairs of setup/hold times can be found by determining the register's output level at a particular time t_f,[2] again for many trial combinations of setup and hold skews, to obtain a surface. This is again followed by contour extraction; all points in the setup/hold time plane for which the output reaches, for example, 50% of the final value at time t_f are found—this contour has a constant clock-to-q delay, which is degraded by 10%.

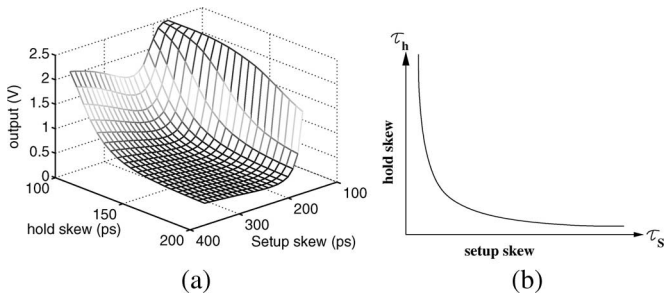

Fig. 1. (a) Q output surface as a function of setup and hold skews. (b) Contour corresponding to a 10% increase in the constant clock-to-q delay.

One such surface for a register output Q versus setup/hold skews is shown in Fig. 1(a); the setup/hold contour obtained is shown in Fig. 1(b). The contour shown in Fig. 1(b) has a constant clock-to-q delay, which is degraded by 10%, hence represents interdependent setup/hold time pairs of interest for timing analysis.

A bottleneck limiting the use of interdependent setup/hold time information in timing analysis flows is the cost of generating constant clock-to-q delay contours, typically obtained by postprocessing output surfaces such as the one shown in Fig. 1(a). Automated generation of output surfaces involves a much larger number of transient simulations than for the already expensive task of characterizing setup and hold times independently of each other.

Again, adapting ideas from mixed-signal/RF simulation and also from homotopy/numerical continuation [5]–[7], [13], [14], we devise a technique for directly finding interdependent setup/hold time contours without the need for generation of output surfaces. The first step in our approach is to formulate the interdependent setup/hold time problem as an underdetermined scalar nonlinear equation $h(\tau_s, \tau_h) = 0$, where τ_s and τ_h are setup and hold skews, respectively. $h(\cdot, \cdot)$ is obtained by computing the nonlinear state-transition function [8], [9] of the differential equations describing the latch or register, as will be discussed in more detail later in this paper. We then use a modified Moore–Penrose pseudoinverse-based Newton–Raphson (MPNR) method [10], [13] to numerically solve the equation $h(\tau_s, \tau_h) = 0$ for *one pair* of interdependent setup/hold times. Once this point is found, our algorithm proceeds to efficiently determine other points on the constant clock-to-q contour by an Euler–Newton (EN) curve-tracing procedure [13], [15].

EN curve tracing operates by using information from a given solution point on the clock-to-q delay curve to efficiently solve for a neighboring point on the curve. It leverages null-space information from the Jacobian matrix of $h(\tau_s, \tau_h)$ to first find a tangent to the curve at the known solution point. It then extrapolates along this tangent to predict a good approximation to a neighboring solution, which it then rapidly[3] refines to any desired accuracy, using the same MPNR nonlinear solution method used to obtain the first solution point. This process is repeated to find the entire constant clock-to-q delay contour.

The key property that makes this curve-tracing procedure more efficient than surface generation is that the computation

[2] t_f is the time at which clock-to-q delay increases by, for example, 10%.

[3] That is, two to three MPNR iterations are typical.

of irrelevant points on the surface is totally avoided. As a result, the number of latch simulations involved in curve tracing is linear in the number of points n desired for characterizing the constant clock-to-q contour, as opposed to $O(n^2)$ for brute-force output surface generation.

Hence, EN curve tracing provides speedups of about n times over output surface generation, where n, which is the number of points on the curve, also constitutes a measure of the precision to which the setup/hold time contour is desired.

In validations of the curve-tracing technique on true single-phased clocked (TSPC) and C^{2}MOS registers (Section VII), we obtain speedups of about 10–12$\times$ for $n = 17-24$ points on the curve. Additionally, the points obtained on the curve by the EN method are "exact" (i.e., refined to any prescribed accuracy by MPNR), whereas the brute-force technique uses interpolation at the postprocessing stage to extract the contour from the output surface. We also provide results on a transmission-gate-based register, which features very sharp transitions in its Q output and results in no appreciable speedup. Our results illustrate that the setup/hold tradeoff curves of different latch architectures have different qualitative properties, which can potentially be exploited at higher levels of timing analysis.

C. Organization of Paper

The remainder of this paper is organized as follows: We first provide further background on the register setup/hold time problem in Section II. In Section III, we develop the new state-transition equation-based formulation, using the nonlinear differential algebraic equations (DAEs) of latches/registers to express the problem of finding setup/hold times as $h(\tau_s, \tau_h) = 0$. Section IV is divided into two subsections: Section IV-A decouples the equation $h(\tau_s, \tau_h) = 0$ for independent setup/hold problem into equations $h(\tau_s) = 0$ and $h(\tau_h) = 0$; Section IV-B presents a detailed description of our NR-based numerical algorithm to individually solve these decoupled equations. In Section V, we validate the new method for independently characterizing setup/hold times on prototypical register structures and provide detailed comparisons against binary search techniques.

Section VI is divided into three subsections: Section VI-A develops the procedure for solving $h(\tau_s, \tau_h) = 0$ by MPNR, focusing on a single point on the curve; Section VI-B outlines the EN method for tracing the solution curve of $h(\tau_s, \tau_h) = 0$; finally, Section VI-C puts together the complete EN curve-tracing algorithm for interdependent setup/hold times using a pseudocode description. In Section VII, we validate the new technique for interdependently characterizing setup/hold times on practical latch/register circuits and compare against brute-force output surface generation, confirming that EN successfully traces the constant clock-to-q delay curve accurately and with large speedups. We also illustrate, using the example of a transmission gate register, the limitations of our method when the Q output is extremely sensitive to setup/hold skews.

II. Terminology and Brief Background

Latches and edge-triggered registers are crucial and ubiquitous building blocks in all digital designs. Typically, each register has a clock line and a data line [16]. For a reliable transfer of data through the register, the data line must be stable for a certain amount of time (known as the *setup time*) prior to the active clock edge. Similarly, the data input line must be held stable for a certain amount of time (known as *hold time*) after the active clock edge as well, to avoid problems in sampling the data. The active clock edge is defined as the transition edge of the clock at which data transfers occur; it is the *low-to-high/high-to-low* transition for a positive-triggered/negative-triggered register.

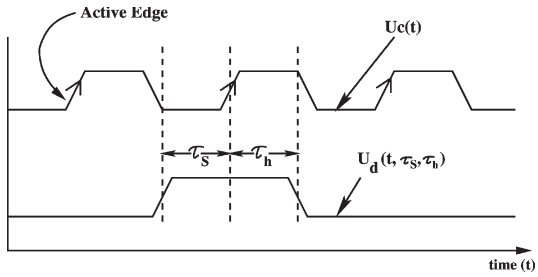

Fig. 2. Clock/data waveforms.

Clock-to-q delay is a term commonly used in the context of latches/registers; it refers to the delay from the 50% transition of the active clock edge to the 50% transition of the Q (output) of the latch/register. Setup skew is the delay from the 50% transition of the data line to the 50% transition of the active clock edge; similarly, hold skew is the delay from the 50% transition of the active clock edge to the 50% transition of the data line. These concepts are illustrated in Fig. 2, where setup and hold skews are denoted by τ_s and τ_h, respectively.

As already noted, setup and hold times are not independent quantities, but can strongly depend on each other. Typically, the setup time decreases as the hold skew increases, and the hold time decreases as the setup skew increases. The tradeoff between setup and hold skews is a strong function of register architecture. Because of this interdependence, a constant clock-to-q delay occurs for many pairs of setup and hold skews, as shown in Fig. 1(b).

III. Nonlinear State-Transition Formulation for Register Setup and Hold Time Determination

In this section, we establish the formulation of finding setup/hold times as an equation and then give a detailed procedure of solving the formulated problem, first, for ignoring the interdependence of setup and hold times and considering them as independent quantities and, second, for taking the interdependence of setup and hold times into account.

A. Setup/Hold Time Problem Formulation

Any nonlinear circuit or system can be represented by the following vector DAE [9]:

$$\frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}(t) = 0. \tag{1}$$

Equation (1) is a size n system; $\vec{x} \in \mathbb{R}^n$ is the state vector of internal node voltages and branch currents; $\vec{q} \in \mathbb{R}^n$ and $\vec{f} \in \mathbb{R}^n$

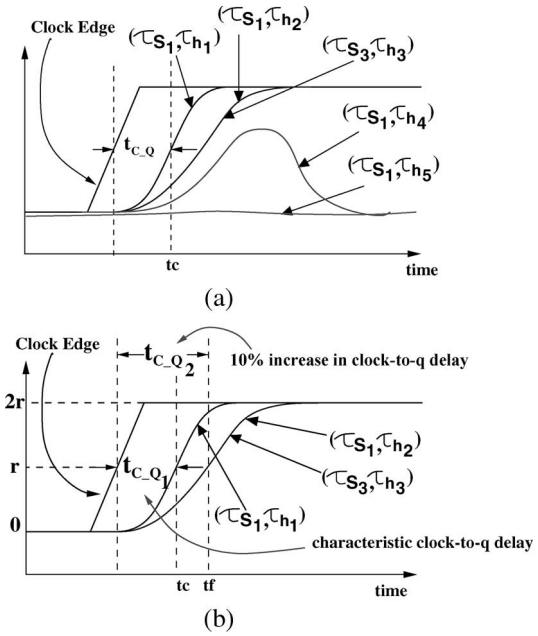

Fig. 3. Behavior of the output waveform for different setup skews. (a) Output for $\tau_{h1} > \tau_{h2} > \tau_{h3} > \tau_{h4} > \tau_{h5}$. (b) Clock-to-$q$ delay.

are the charge/flux and the current terms, respectively; and $\vec{b}(t) \in \mathbb{R}^n$ represents all the input source voltages and currents.

A typical register consisting of many transistors has a clock line and one or more data lines as input. It is a nonlinear circuit system and can be represented by (1).

Without loss of generality, we assume positive-edge-triggered registers and denote the clock waveform by $u_c(t)$ and the data waveform by $u_d(t, \tau_s, \tau_h)$, as shown in Fig. 2. τ_s and τ_h represent setup and hold skews, respectively. Clearly, the shape of the data waveform depends on τ_s and τ_h; therefore, it is denoted as $u_d(t, \tau_s, \tau_h)$ to explicitly bring out its dependence on τ_s and τ_h.

Once the clock and data inputs (refer to Fig. 2) are separated as aforementioned, the differential equations of the register can be written as follows:

$$\frac{d}{dt}\vec{q}(\vec{x}(t,\tau_s,\tau_h)) + \vec{f}(\vec{x}(t,\tau_s,\tau_h)) + \vec{b}_c u_c(t) + \vec{b}_d u_d(t,\tau_s,\tau_h) = 0. \quad (2)$$

To detect a pair of setup and hold skews (τ_s, τ_h) for which the clock-to-q delay increases by 10%, we need to monitor an output waveform, which is given by $\vec{c}^T\vec{x}$. Here, $\vec{c}$ will typically be a unit vector that selects an output node. The typical behavior of the output waveform for different values of τ_s and τ_h is shown in Fig. 3(a). Note from Fig. 3(a) that for a constant value of setup skew τ_{s1}, the clock-to-q delay increases as the hold skew τ_h decreases. Note also that two pairs of different setup and hold skews (τ_{s1}, τ_{h2}) and (τ_{s3}, τ_{h3}) can result in the same clock-to-output delays. Typically, if setup and hold skews both become larger than certain threshold values, the clock-to-q delay becomes independent of setup and hold skews, approaching the *characteristic clock-to-q delay* of the register.

We are interested in finding those pairs of setup and hold skews (τ_s, τ_h) for which the clock-to-q delay increases by 10% from the characteristic clock-to-q delay—this is a typical criterion for defining setup and hold times. Let t_c denote the time at which the output reaches the characteristic clock-to-q delay; t_f, the time when the clock-to-delay increases by 10%; and r, the value of the output at the 50% transition. These quantities are graphically depicted in Fig. 3(b)—t_{C-Q1} represents the characteristic clock-to-q delay, and t_{C-Q2} represents a 10% increase in t_{C-Q1}. The setup/hold skew pairs that lead to a 10% degradation in the clock-to-q delay in the second waveform are defined to be setup and hold times.

Let the state transition matrix of (2) be denoted by $\vec{\phi}(t; \vec{x}_0, t_0 = 0, \tau_s, \tau_h)$, and let the initial condition $\vec{x}_0 = \vec{x}(t = t_0)$ be fixed to a given value. The setup/hold time determination problem consists of seeking (τ_s, τ_h), given r, t_f, and $\vec{x}_0$, such that the output is at value r at time t_f, i.e., $\vec{c}^T\vec{x}(t_f) = r$. Writing this in terms of the state transition function, we obtain

$$\vec{c}^T\vec{\phi}(t_f; \vec{x}_0, 0, \tau_s, \tau_h) - r = 0 \quad \text{or} \quad \vec{c}^T\vec{\phi}_\tau(\tau_s, \tau_h) - r = 0 \quad (3)$$

where $\vec{\phi}_\tau(\tau_s, \tau_h) \equiv \vec{\phi}(t_f; \vec{x}_0, 0, \tau_s, \tau_h)$. Hence, the nonlinear equation that we need to solve to obtain (τ_s, τ_h) is

$$h(\vec{\tau}) \equiv h(\tau_s, \tau_h) \equiv \vec{c}^T\vec{\phi}_\tau(\tau_s, \tau_h) - r = 0 \quad (4)$$

where $\vec{\tau} = [\tau_s, \tau_h]$.

IV. NR-Based Independent Characterization

A. Specialization for Independent Characterization

Note that a simplification of (4) can be used to find either the setup or the hold time by assuming a fixed unchanging value for the other. For example, for a fixed value of τ_h, (4) can be written as follows [17]:

$$h(\tau_s) \equiv \vec{c}^T\vec{\phi}_\tau(\tau_s) - r = 0. \quad (5)$$

Equation (5) is a scalar equation with one scalar unknown τ_s; hence, it can be solved using the standard NR method [10].[4]

To solve it via NR, we need to be able to do two things: 1) evaluate $h(\tau)$ given any τ and 2) evaluate $dh(\tau)/d\tau$ for any τ. Evaluation of $h(\tau)$ can be done by running a transient simulation with the given τ and then evaluating (5). $dh(\tau)/d\tau$ can be computed by performing sensitivity analysis, which is a well-established method [11], [12], on (2). The direct sensitivity method adapted to compute $dh(\tau)/d\tau$ is given in Appendix A.

B. NR for Independent Setup/Hold Time Characterization

In this section, we outline the application of the NR algorithm to solve (5), as follows:

1) Initialize τ, $\vec{x}$, and $\vec{m}$ (see Appendix A for the definition of $\vec{m}$).
   a) Set $\tau = \tau_0$.
      τ_0 is an initial guess for setup (or hold) time.

[4]For further analysis of (5), we will drop the subscript "s" from τ_s and, similarly, for hold time.

b) $\vec{x}(t=0,\tau_0)=\vec{x}_0(\tau_0)$.
This is the initial condition from which the simulation starts; starting from the dc operating point is typical.
c) $\vec{m}(t=0,\tau_0)=\vec{m}_0(\tau_0)$.
As explained in Appendix A, $\vec{m}_0(\tau_0)$ can be initialized to $\vec{0}$ if the initial condition $\vec{x}_0$ does not depend on the setup (or hold) skew; this is typical.

2) Start the NR procedure. At the jth iteration:
a) Divide the interval $t=0$ to t_f in N points: t_0, t_1, $\ldots, t_{N-1}$. For each $i \in \{0,\ldots,N-1\}$, compute the following:
τ_j is the value of τ being used for the iteration index j.
i) Compute $\vec{x}_{ji}$ using (21).
Here, $\vec{x}_{ji}$ denotes the fact that the quantity $\vec{x}$ is being evaluated at time t_i for the iteration index j. This terminology will also hold for other quantities. Equation (21) can be solved using any integration method, such as BE, TRAP, etc. [8], [9].
ii) After having obtained $\vec{x}_{ji}$'s, compute the following:

$$C_{ji}=\left.\frac{d\vec{q}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{ji}} \quad \text{and} \quad G_{ji}=\left.\frac{d\vec{f}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{ji}}. \tag{6}$$

iii) Compute $\vec{m}_{ji}$ using (26) as follows:

$$\vec{m}_{ji}=\left(\frac{C_{ji}}{t_i-t_{i-1}}+G_{ji}\right)^{-1} \times\left(\frac{C_{j(i-1)}}{t_i-t_{i-1}}\vec{m}_{j(i-1)}-\vec{b}_d u_d'(t_i,\tau_j)\right). \tag{7}$$

We have now obtained $\vec{x}_{j(N-1)}$ and $\vec{m}_{j(N-1)}$.
b) Calculate $h(\tau_j)$ defined in (5) as follows:

$$h(\tau_j)=\vec{c}^T\vec{x}_{j(N-1)}-r. \tag{8}$$

c) Check the convergence of NR using reltol and abstol [18].
If NR has converged, then we have obtained the optimal value of τ as τ_j. Stop here.
Otherwise, calculate $dh(\tau)/d\tau$ defined in (27) as follows:

$$\left.\frac{dh(\tau)}{d\tau}\right|_{\tau=\tau_j}=\vec{c}^T\vec{m}_{j(N-1)}. \tag{9}$$

d) Calculate τ_{j+1} and increment j, i.e.,

$$\tau_{j+1}=\tau_j-\frac{h(\tau_j)}{\left.\frac{dh(\tau)}{d\tau}\right|_{\tau=\tau_j}} \quad \text{and} \quad j=j+1. \tag{10}$$

Go to step (a) for the next iteration of NR.

The previously described procedure will be applied for the computation of τ_s in the next section.

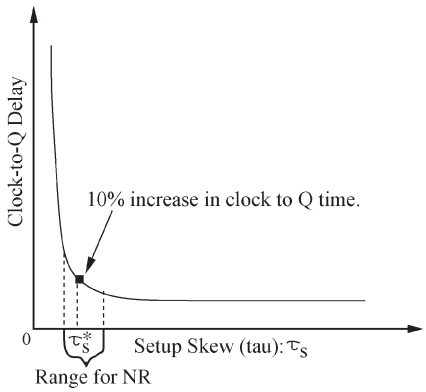

Fig. 4. Setup skew versus clock-to-output delay.

V. Application of NR Methodology for Independent Setup/Hold Time Characterization and Results

In this section, we will first review the existing method for the *independent* setup/hold time characterization. We will then show how NR is useful for fast and more precise computation of the setup time. At the end of this section, we will compare the results obtained from the NR method against the binary search method to emphasize the usefulness of NR-based methodology for latch/register characterization.

In one of the currently existing methodology for the *independent* setup time characterization, the clock-to-output delay is first plotted against the setup skew [a typical plot of t_{C-Q} versus setup skew is shown in Fig. 4]; then, the setup skew for which the clock-to-q delay increases by a certain amount, e.g., by 10%, is measured and considered as the setup time. We must note here that the clock-to-q delay changes very rapidly near the setup time, making it very sensitive to small setup skew changes in the neighborhood of the setup time; therefore, capturing the setup time with high accuracy requires many simulations to run in the vicinity of the setup time τ_s^*.

If we have identified the region around the setup time, i.e., we have found two nearby setup skews such that it contains τ_s^*, then we can refine the search for the setup time (using the criteria of a 10% increase in t_{C-Q}) by applying the binary search method in that interval. One such interval is shown in Fig. 4, which contains the setup time point τ_s^*.

We propose here to apply the NR method in the previously identified setup skew interval to quickly narrow down to the setup time. Since NR has quadratic convergence near the solution, we expect to get to the result much faster as compared to the binary search method, which has linear convergence.

A. Transmission-Gate-Based Master–Slave Register

We have chosen the simple transmission gate master–slave positive-edge-triggered register for the verification purpose. The transmission gate master–slave register is shown in Fig. 5.

The clock waveform $(u_c(t))$ used in the register is chosen such that it makes the transition between 0 and 2.5 V with a period of 10 ns. It has a rise/fall time of 0.1 ns and has an initial delay of 1 ns. Therefore, the active clock transition edge starts at 1 ns, 11 ns, 21 ns, etc. The data waveform $(u_d(t,\tau_s))$ is centered

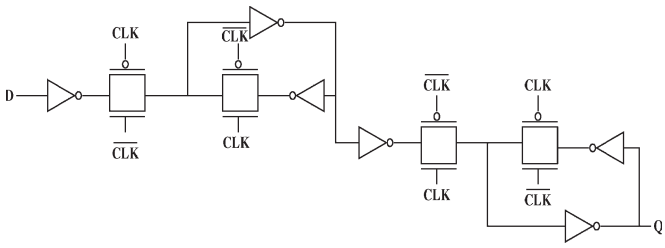

Fig. 5. Transmission-gate-based positive-edge-triggered master–slave register.

around the active clock edge, which starts at 11 ns. τ_h has been kept fixed, and therefore, the data waveform changes its shape depending on the value of τ_s.

At first, to determine the nominal (characteristic) clock-to-q delay, the register is simulated for a very large value of τ_s, and the output is monitored. The output waveform rises to its 50% value, i.e., 1.25 V at $t_c = 11.625$ ns; hence, the nominal t_{C-Q} is equal to 575 ps. The clock-to-q delay corresponding to a 10% increase in the nominal t_{C-Q} becomes equal to 632.5 ps, and it allows us to set $t_f = 11$ ns + (rise time/2) + 632.5 ps = 11.682 ns and $r = 1.25$ V in (5) [t_c, t_f, and r used here correspond to the same symbols in Fig. 3(b)].

After having set the value for r and t_f, we need a suitable initial guess of τ_s to start the NR method. The initial guess of τ_s proves to be critical for the convergence of NR and should be chosen near the solution within the convergence range of NR.

To find a good initial guess of τ_s, we first start with a setup skew interval $[\tau_{sL}, \tau_{sR}]$, where the register properly latches the data for τ_{sL} and fails to latch for τ_{sR}. Therefore, this interval will contain τ_s^*. We then narrow down the setup skew interval surrounding τ_s^* [as shown in Fig. 4] using the binary search method until the interval length falls in the convergence range of NR.

The convergence range of NR varies with the types of registers, the rise/fall time of data and clock waveforms, etc. The sensitivity of the output waveform to the changes in the setup skew considerably affects the convergence range of NR. The convergence range will be small for a more sensitive output and large for a less sensitive output.

For the circuit in Fig. 5, NR has a very small convergence range, approximating to 5–8 ps. Therefore, we narrowed down the setup skew interval to 5 ps before applying NR on it. We can then take the initial guess of τ_s either as τ_{sR} or τ_{sL} to start the NR method. After having the values for τ_{s0} (initial guess), r, and t_f, we apply NR (following the algorithm described in the previous section) to reach the solution τ_s^*.

We first calculate the solution τ_s^* using NR that is accurate up to double precision and use this result as a baseline to estimate errors in the solution (τ_s^*) of lower digits of accuracy, which is obtained either by NR or by the binary search method. We then plot the number of iterations needed in the NR and binary search methods against the percentage of error in the solution relative to double-precision accuracy.

The plots in Fig. 6(a) show the number of iterations needed by NR to reduce the error by about an order of magnitude and compare this number with those needed by binary search. As expected, the NR procedure requires fewer iterations.

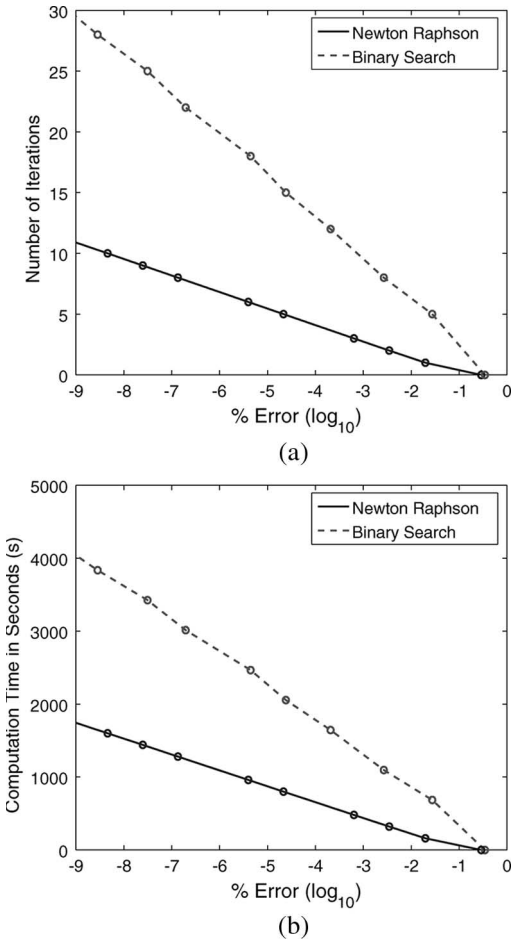

Fig. 6. %Error $(\log_{10})$: NR versus binary search method. (a) Number of iterations versus %Error $(\log_{10})$. (b) Computation time versus %Error $(\log_{10})$.

The computation time per iteration of NR is typically more than the binary search due to the evaluation of (7) in each iteration. Equation (7) involves the inverse computation of matrix $((C_{ji}/t_i - t_{i-1}) + G_{ji})$ and appears to be expensive in terms of time consumption. However, effectively, the inverse computation of this matrix is of the order $O(n^a)$ $(a \in [1, 2])$ because of the sparsity.

For the circuit shown in Fig. 5, NR took 160.5 s per iteration, whereas binary search took 137.2 s. The plots shown in Fig. 6(b) give the comparison of computation time for the NR and binary search methods. Clearly, NR consumes less time than binary search for an equal amount of reduction in the error, in spite of its larger computation time per iteration. The vertical difference between the plots shown in Fig. 6(b) becomes bigger and bigger, validating our assumption that NR will perform better with each additional bit of accuracy.

B. C^2MOS Positive-Edge-Triggered Master–Slave Register

To further validate the use of NR as a fast characterization method, we extracted the setup time information for the circuit shown in Fig. 7 using NR. We have taken the same clock and data waveform ($u_c(t)$ and $u_d(t, \tau_s)$) as used for the previous register. The output of this register rises to its 50% value, i.e., 1.25 V at 11.171 ns, for a large setup skew. As we did in

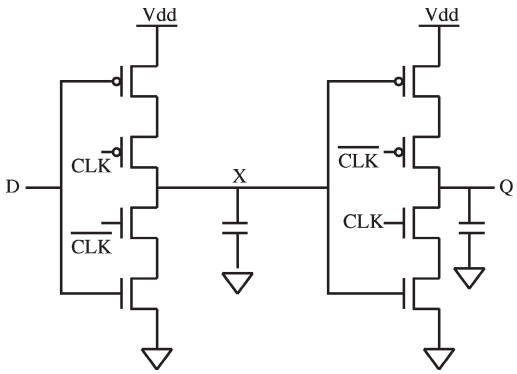

Fig. 7. C^2MOS positive-edge-triggered master–slave register.

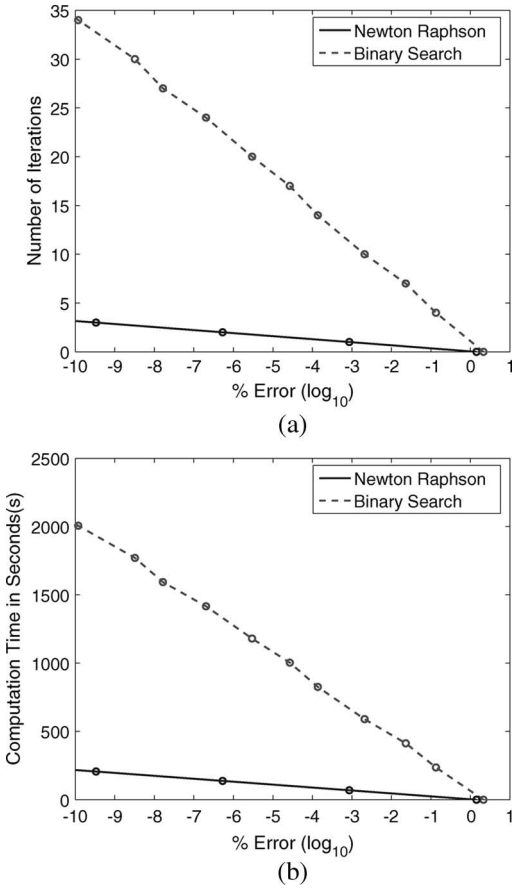

Fig. 8. %Error $(\log_{10})$: NR versus binary search method. (a) Number of iterations versus %Error $(\log_{10})$. (b) Computation time versus %Error $(\log_{10})$.

the previous example, we set $r = 1.25$ V and $t_f = 11.183$ ns in (5).

The convergence range of NR for this register is found to be approximately 120–150 ps, indicating that the output of this register is less sensitive to the changes in setup skews as compared to the output of the transmission-gate-based master–slave register. Therefore, the initial value of τ_s for this register can be chosen anywhere within the setup skew interval of length 120–150 ps containing τ_s^* (*setup time*). After having the numerical values for r, t_f, and τ_0, we apply NR and first compute the setup time of the register that is accurate up to 16 digits for estimating the errors in the solution of lower digits of accuracy. The plots shown in Fig. 8(a) and (b) show the number of iterations and computation time against the percentage of error in the solution, relative to double-precision accuracy. Once again, it is evident from the plots that NR becomes more and more efficient than binary search with every incremental reduction (on a logarithmic scale) in the error of the solution.

Our results clearly suggest that NR is useful only for higher accuracies, and one may wonder about how many relevant digits are actually needed in the value of the setup/hold time during the characterization process. Since the convergence range of NR forces the solution to be accurate up to two to three digits of accuracy to begin the NR process to obtain a more accurate solution, the proposed method is not useful when one needs only up to three to four digits of accuracy. However, in general, a computer-aided design (CAD) company typically characterizes latches to about seven digits of accuracy; hence, we believe that the NR method will be of practical importance in the industrial characterization flows.

The usefulness of NR will also be more prominent if we already have some ideas of the setup skew interval surrounding τ_s; then, it only requires us to run few iterations of NR to extract the setup time of high accuracy. It turns out that we, in fact, have a fair idea of such setup skew interval from the previous characterization of the standard cell library containing similar registers. Hence, NR can be less time consuming and, thus, very useful as it features more accuracy in less number of iterations, in the characterization of the setup/hold time.

VI. Interdependent Setup/Hold Time Characterization

We have already established in the previous section how setup/hold times can be found using the simple NR method with higher accuracy when their interdependence is ignored. In this section, we take the interdependence of setup and hold times into consideration and show how to solve the underdetermined scalar (4) using a Moore–Penrose pseudoinverse Newton procedure. We then develop an EN curve-tracing procedure for this equation.

A. Solving for Interdependent Setup/Hold Times via MPNR

Since (4) is an underdetermined scalar nonlinear equation with two unknowns τ_s and τ_h, NR, which is the method of choice for the numerical solution of "square" nonlinear systems, cannot be directly applied. However, a modification, which is based on the application of the Moore–Penrose pseudoinverse [13] during the linear solution step, can be applied instead, as described in this section. Intuitively, MPNR starts with an initial guess of (τ_{s0}, τ_{h0}) and converges to a solution (τ_s^c, τ_h^c) of (4), which lies on the constant clock-to-q delay curve, as shown in Fig. 9. A denotes the initial guess, whereas B denotes the point on the solution curve (Fig. 9) that MPNR converges to. It can be proven that under the right circumstances, MPNR will converge to a point B on the solution curve that is closest to A.

To solve (4) using MPNR, it is necessary to perform three tasks: 1) evaluate $h(\tau_s, \tau_h)$ given any (τ_s, τ_h); 2) evaluate $[dh(\vec{\tau})/d\vec{\tau}] = [dh(\vec{\tau})/d\tau_s, dh(\vec{\tau})/d\tau_h]$, which is a 1×2

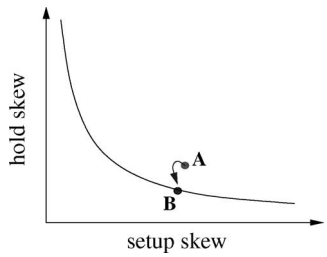

Fig. 9. Convergence of a point via NR on a constant clock-to-q delay curve.

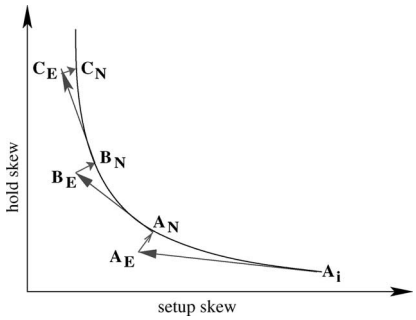

Fig. 10. Curve tracing using the EN method.

matrix; and 3) compute the Moore–Penrose pseudoinverse of $[dh(\vec{\tau})/d\vec{\tau}]$.

$h(\tau_s, \tau_h)$ is simply evaluated by running a transient simulation with the given (τ_s, τ_h) and then evaluating (4). Computation of $[dh(\vec{\tau})/d\vec{\tau}]$ is done by sensitivity analysis [11], [12] on (2). Refer to Appendix B for the detailed description of computing $[dh(\vec{\tau})/d\vec{\tau}]$.

Finally, denoting the matrix $[dh(\vec{\tau})/d\vec{\tau}]$ by $H(\vec{\tau})$, its Moore–Penrose pseudoinverse [13] can be expressed as

$$H(\vec{\tau})^{+}\Big|_{(\tau_s,\tau_h)=\left(\tau_s^*,\tau_h^*\right)} = H(\vec{\tau})^{t}\left(H(\vec{\tau})H(\vec{\tau})^{t}\right)^{-1}\Big|_{(\tau_s,\tau_h)=\left(\tau_s^*,\tau_h^*\right)} \tag{11}$$

where $H(\vec{\tau})^{+}$ and $H(\vec{\tau})^{t}$ represent the *pseudoinverse* and *transpose* of matrix $H(\vec{\tau})$, respectively.

The MPNR algorithm is given in detail in Section VI-C.

B. Constant Clock-to-q Delay Contour Tracing by EN

In the previous section, we provided the key computational details of the MPNR procedure, which is used to find a single point on a constant clock-to-q delay curve when an initial guess of setup and hold skews is given. In this section, we outline an EN-based method for tracing the entire constant clock-to-q contour in the $\tau_s - \tau_h$ plane [as shown in Fig. 1(b)], i.e., the set of all solutions of (4).

The EN curve-tracing [13] method used here follows a standard predictor–corrector methodology [8], using Euler steps as predictors and MPNR steps as correctors. Taking an Euler predictor step involves computing the tangent vector to the solution curve at a previously known point on the curve and extrapolating to a new point along the tangent. The MPNR procedure is then used as a corrector that uses this new point as its initial guess and converges to a nearby solution point on the curve. The EN curve-tracing procedure is graphically depicted in Fig. 10, where the blue and red arrows denote the Euler predictor steps and the MPNR corrector steps, respectively.

The previous section has already provided details regarding the computation of all quantities needed to perform MPNR. For the Euler step, the key quantity we need to evaluate is a unit tangent vector at any given point (τ_s^c, τ_h^c) on the curve. The unit tangent vector at point $\vec{\tau}^c = (\tau_s^c, \tau_h^c)$, denoted by $T(H(\vec{\tau}))$ (and called "the tangent vector induced by $H(\vec{\tau})$"), can be computed as follows [13]:

$$T\left(H(\vec{\tau})\right)\big|_{\vec{\tau}=\vec{\tau}^c} = \begin{pmatrix} \frac{-dh(\vec{\tau})}{d\tau_h} \\ \frac{dh(\vec{\tau})}{d\tau_s} \end{pmatrix} \frac{1}{\sqrt{\left(\frac{dh(\vec{\tau})}{d\tau_s}\right)^2 + \left(\frac{dh(\vec{\tau})}{d\tau_h}\right)^2}}\Bigg|_{\vec{\tau}=\vec{\tau}^c}. \tag{12}$$

The rectangular matrix in (12) is simply the MPNR Jacobian matrix at the current solution point; hence, it is already available; since it is of size 2, the computation involved in finding the tangent vector is trivial.

C. Euler–MPNR-Based Curve-Tracing Algorithm for Interdependent Setup/Hold Time Characterization

We now outline the overall procedure of finding all the pairs (τ_s^c, τ_h^c) such that they satisfy (4). Equation (4) needs to be evaluated at time $t = t_f$. We again note that the curve obtained as the solution set of (4) will be a constant clock-to-q delay curve in the plane of setup and hold skews, and therefore, each point on the curve represents a setup/hold time pair.

The complete algorithm for tracing the constant clock-to-q delay curve is shown in the following.

Note: Superscript "c" will be used to denote quantities that lie on the constant clock-to-q delay curve.

1) Initialize τ_s, τ_h, $\vec{x}$, $\vec{m}_s$, and $\vec{m}_h$ (see Appendix B for the definition of $\vec{m}_s$ and $\vec{m}_h$).
   a) $(\tau_s, \tau_h) = (\tau_{s0}, \tau_{h0})$.
      τ_{s0} and τ_{h0} are initial guesses for setup and hold times, respectively.
   b) $\vec{x}(t = 0, \vec{\tau}) = \vec{x}_0(\vec{\tau})$.
      This is the initial condition from which the simulation starts; starting from the dc operating point is typical.
   c) $\vec{m}_s(t=0, \vec{\tau}_0) = \vec{m}_{s0}(\vec{\tau}_0)$ and $\vec{m}_h(t=0, \vec{\tau}_0) = \vec{m}_{h0}(\vec{\tau}_0)$.
      As explained in Appendix B, $\vec{m}_{s0}(\vec{\tau}_0)$ and $\vec{m}_{h0}(\vec{\tau}_0)$ can be initialized to $\vec{0}$ if the initial condition $\vec{x}_0$ does not depend on the setup and hold skews; this is typical.
2) Start the EN procedure.
   For an Euler iteration index k.
   Note: $k-1$ essentially represents the number of points that has already been computed on the curve.
   a) Start the NR procedure. For an iteration index j inside NR:
      i) Divide $t = 0$ to t_f in N points: $t_0, t_1, \ldots, t_{N-1}$. For each $i \in \{0, \ldots, N-1\}$, compute the following:
      $\vec{\tau}_{kj} = [\tau_{skj}, \tau_{hkj}]$ is the value of $\vec{\tau}$ being used for the NR iteration index j and Euler index k.

A) Compute $\vec{x}_{kji}$ using (2).
Here, $\vec{x}_{kji}$ denotes the fact that the quantity $\vec{x}$ is being evaluated at time t_i for the NR iteration index j and Euler index k. This terminology will also hold for other quantities. Equation (2) can be solved using any integration method, such as BE, TRAP, etc. [8], [9].

B) After having obtained $\vec{x}_{kji}$'s, compute the following:

$$C_{kji} = \left.\frac{d\vec{q}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{kji}} \quad \text{and} \quad G_{kji} = \left.\frac{d\vec{f}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{kji}}. \quad (13)$$

C) Compute $(\vec{m}_s)_{kji}$ using (32) as follows:

$$(\vec{m}_s)_{kji} = \left(\frac{C_{kji}}{t_i - t_{i-1}} + G_{kji}\right)^{-1} \times \left(\frac{C_{kj(i-1)}}{t_i - t_{i-1}}(\vec{m}_s)_{kj(i-1)} - \vec{b}_d z_s(t_i, \vec{\tau}_{kj})\right). \quad (14)$$

Compute $(\vec{m}_h)_{kji}$ similarly.

We have now obtained $\vec{x}_{kj(N-1)}$, $(\vec{m}_s)_{kj(N-1)}$, and $(\vec{m}_h)_{kj(N-1)}$.

ii) Calculate $h(\vec{\tau}_{kj})$ defined in (4) as follows:

$$h(\vec{\tau}_{kj}) = \vec{c}^T \vec{x}_{kj(N-1)} - r. \quad (15)$$

iii) Check the convergence of NR using reltol and abstol [19]. If NR has converged, then we have obtained the optimal value of $\vec{\tau}$ as $\vec{\tau}_{kj}$. Jump to step (b) to compute the Euler step at $\vec{\tau}_{kj}$.
Also, we will denote this optimal value of $\vec{\tau}_{kj}$, which essentially lies on the curve, as $\vec{\tau}^c_k$ outside the NR procedure.
Otherwise, calculate $dh(\vec{\tau})/d\tau$ defined in (33) as follows:

$$\left.\frac{dh(\vec{\tau})}{d\tau_s}\right|_{\vec{\tau}=\vec{\tau}_{kj}} = \vec{c}^T(\vec{m}_s)_{kj(N-1)}$$
$$\left.\frac{dh(\vec{\tau})}{d\tau_h}\right|_{\vec{\tau}=\vec{\tau}_{kj}} = \vec{c}^T(\vec{m}_h)_{kj(N-1)}. \quad (16)$$

iv) Calculate $\tau_{k(j+1)}$ and increment j. Therefore

$$\tau_{k(j+1)} = \tau_{kj} - h(\vec{\tau}_{kj}) \left(\frac{dh(\vec{\tau})}{d\tau}\right)^+ \Bigg|_{\vec{\tau}=\vec{\tau}_{kj}} \quad \text{and} \quad j = j+1. \quad (17)$$

In (17), $(dh(\vec{\tau})/d\tau)^+$ is the Moore–Penrose pseudo-inverse of $dh(\vec{\tau})/d\tau$, which can be calculated using (11) as follows:

$$\left(\frac{dh(\vec{\tau})}{d\tau}\right)^+ = \left(\frac{dh(\vec{\tau})}{d\tau}\right)^t \left[\frac{dh(\vec{\tau})}{d\tau}\left(\frac{dh(\vec{\tau})}{d\tau}\right)^t\right]^{-1} \quad (18)$$

where $(dh(\vec{\tau})/d\tau)^t$ is the transpose of $dh(\vec{\tau})/d\tau$.

Go to step (i) for the next iteration of NR.

b) Compute the following tangent unit vector induced by $dh(\vec{\tau})/d\vec{\tau}$ using (12) for $\vec{\tau} = \vec{\tau}^c_k$:

$$T\left(\frac{dh(\vec{\tau})}{d\tau}\right) = \begin{pmatrix} \frac{-dh(\vec{\tau})}{d\tau_h} \\ \frac{dh(\vec{\tau})}{d\tau_s} \end{pmatrix} \frac{1}{\sqrt{\left(\frac{dh(\vec{\tau})}{d\tau_s}\right)^2 + \left(\frac{dh(\vec{\tau})}{d\tau_h}\right)^2}}. \quad (19)$$

c) Compute the new pair of (τ_{s0}, τ_{h0}) along the unit tangent vector: (predictor step)

$$\vec{\tau}_{(k+1)0} = \vec{\tau}^c_k + \alpha . T\left(\frac{dh(\vec{\tau})}{d\tau}\right)\Bigg|_{\vec{\tau}=\vec{\tau}^c_k} \quad \text{and} \quad k = k+1 \quad (20)$$

where α is the step used in the direction of tangent. Go to step (a) of the EN procedure and keep repeating until the traversing is stopped.

VII. Application of Euler–Newton Methodology for Interdependent Setup/Hold Time Characterization and Results

In this section, we first validate the interdependent setup/hold time characterization algorithm developed in this paper using two types of registers: 1) a TSPC register and 2) a C^{2}MOS positive-edge-triggered master–slave register. We describe the validation procedure in some details. Our results, which we compare against brute-force output surface generation, confirm that the new curve-tracing method accurately finds constant clock-to-q contours in computation linear in the desired number of contour points. We obtain speedups of about 10–12× over surface generation[5] for 17–24 curve points (representing excellent precision for timing analysis purposes). Additionally, we have chosen reltol/abstol for MPNR such that the points obtained on the curve are accurate up to five digits.

We then present a case using a static transmission-gate-based register exhibiting a very sharp behavior in the Q output, where EN methodology does not feature any speedup against the brute-force method. This example illustrates a limitation of our proposed EN method in terms of efficiency, which is the main attraction of this method.

Our current implementation uses a fixed Euler step length α in (20). α is analogous to the time step in transient simulation; as such, it can be made to change dynamically, depending on the curvature of the curve, sensitivity of the Q output with respect to (w.r.t.) setup/hold times (i.e., the rate of Newton convergence), etc., for more efficient traversal of the curve. Such improvements will improve the efficiency of EN-based characterization as compared to the brute-force technique.

[5]Speedup numbers are obtained via apples-to-apples comparisons on an AMD Athlon 64 3000+-based PC, with 512-MB RAM, running Linux kernel 2.6.12. All algorithms are implemented in a MATLAB/C/C++ simulation prototyping environment.

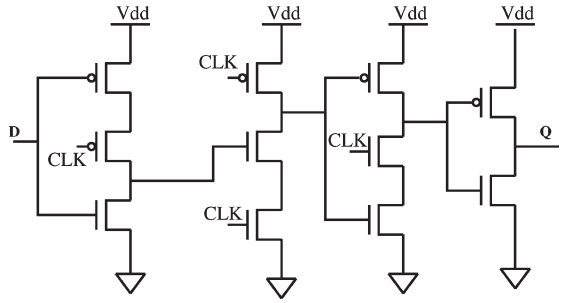

Fig. 11. Positive-edge-triggered register in TSPC.

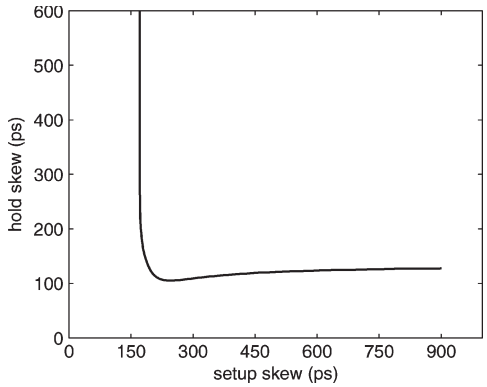

Fig. 12. Constant clock-to-q delay curve obtained by the EN method.

A. TSPC Register

The positive-edge-triggered TSPC register shown in Fig. 11 features positive setup and hold time constraints. The clock waveform $u_c(t)$ used in the register has a period of 10 ns, with logic 0 at 0 V and logic 1 at 2.5 V. The clock input has an initial delay of 1 ns; rise/fall times are both 0.1 ns. Therefore, its active clock edges are located at 1 ns, 11 ns, 21 ns, etc. The chosen data waveform $u_d(t, \tau_s, \tau_h)$ is centered around the active clock edge, which starts at 11 ns. The data waveform changes its shape (variable pulse width, refer to Fig. 2) depending on the values of τ_s and τ_h.

At first, to determine the characteristic clock-to-q delay, the register is simulated for very large values of τ_s and τ_h, and the output is monitored. The output waveform reaches 1.25 V (50% of its final value) at $t_c = 11.348$ ns; hence, the characteristic clock-to-q delay equals 298 ps (the distance from the 50% active clock transition to the 50% transition of the output). Here, we use a standard definition for setup and hold times, which led to an increase of 10% over the characteristic clock-to-q delay. Hence, the constant clock-to-q delay value for contour generation equals 327.8 ps. Accordingly, we set $t_f = 11$ ns + (rise time/2) + 327.8 ps = 11.3778 ns and $r = 1.25$ V in (4) [t_c, t_f, and r used here correspond to the same symbols in Fig. 3(b)].

Determination of the first point on the curve involves starting with a good guess for (τ_{s0}, τ_{h0}) to seed NR. To find one, we make the hold skew τ_{h0} very large, so that the setup time will become largely independent of the hold skew. We start with a setup skew interval $[\tau_{sL}, \tau_{sR}]$, where the register properly latches the data for τ_{sL} and fails to latch data for τ_{sR}. Hence, this interval will contain the setup time point τ_s^c. We then narrow down the setup skew interval using a coarse binary search, until the interval length falls in the convergence range of NR.

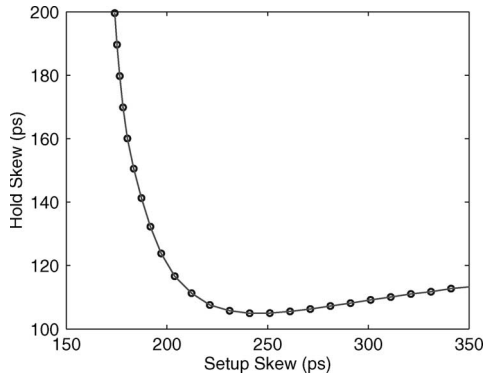

Fig. 13. Constant clock-to-q delay curve obtained by the EN method. The curve represents the pairs of setup and hold skews, which increases the characteristic clock-to-q delay by 10%.

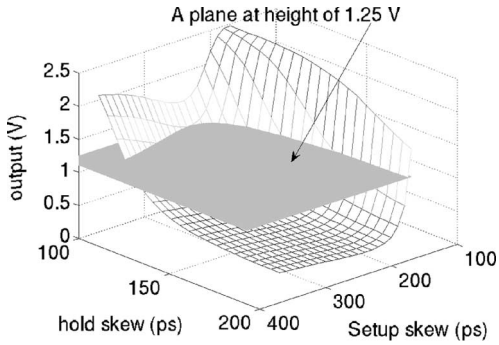

Fig. 14. Q output surface as a function of independent setup and hold skews.

Either τ_{sL} or τ_{sR} can be used as the initial guess τ_{s0} for MPNR solution. Then, we start the EN process, as previously outlined. MPNR typically converges very quickly (two to three iterations) as the curve is traced since the Euler steps provide excellent initial guesses. The constant clock-to-q delay contour obtained by the procedure is shown in Fig. 12. The curve in Fig. 12 flattens out at the extreme edges. To understand its behavior in the middle corner, which can be exploited for setup/hold tradeoffs, we have plotted a part of this curve in Fig. 13.

To verify the correctness of this curve, we also extract the 10% degraded constant clock-to-q delay curve for this register using the brute-force output surface generation technique. An output surface is generated at time $t_f = 11.778$ ns by independently varying setup and hold skews, as shown in Fig. 14. A plane at a "height" of 1.25 V is then drawn to obtain the intersection contour of the output surface. This intersection curve represents the set of all setup/hold skew pairs, which results in a 10% increase in the clock-to-q delay. The top view of the curve obtained by this intersection procedure is shown in Fig. 15. The contour from EN curve tracing (Fig. 13) is overlaid on the intersection of the plane and the output surface in Fig. 15. It is apparent from Fig. 15 that the curve obtained by EN methodology exactly matches the constant clock-to-q delay curve obtained from the surface, thereby verifying the correctness of the new method.

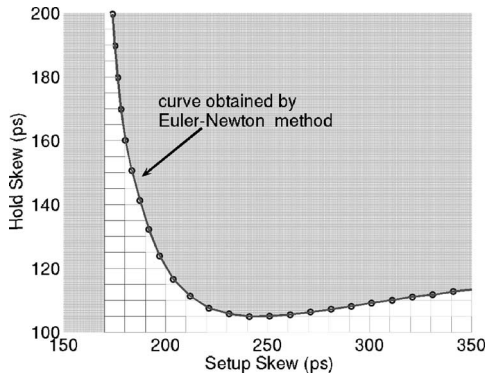

Fig. 15. Top view of the intersection curve of the plane and the output surface and the superimposition of the curve obtained by the EN method.

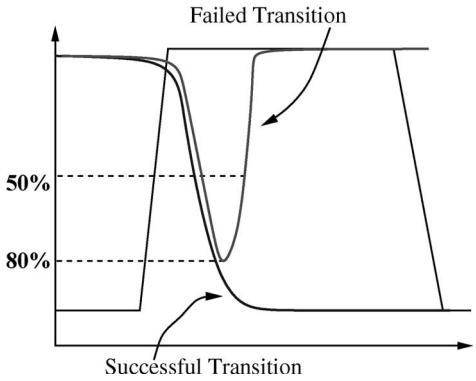

Fig. 16. Output waveform fails to complete the transition even after reaching 80% of its true value.

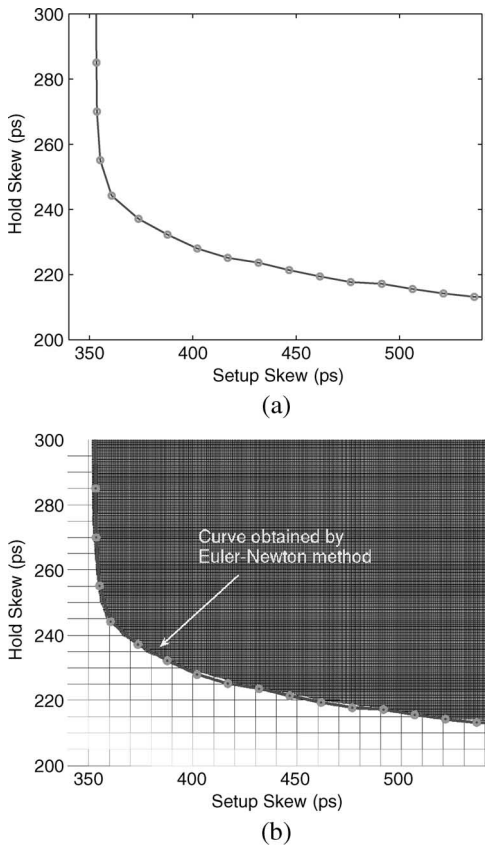

Fig. 17. (a) Contour for a 10% increase in the constant clock-to-q delay, as obtained by EN curve tracing. (b) EN curve overlaid on the intersection curve of the plane and the output surface (top view).

In our implementation, EN curve tracing took 30 min to trace 24 points on the contour, whereas brute-force output surface generation consumed 6 h 14 min (for 24 $\times$ 24 simulations) to obtain 24 points on the curve, representing a speedup of about 12.5$\times$.

B. C^2MOS Positive-Edge-Triggered Master–Slave Register

To further validate the EN curve-tracing method, we apply it to the C^2MOS register shown in Fig. 7. The clock $u_c(t)$ and the data $u_d(t, \tau_s, \tau_h)$ used here are the same as for the previous register. The register shown in Fig. 7 has zero hold time if there is no overlap between the clk and $\overline{clk}$ inputs. To obtain a positive hold time for this register, we delay the $\overline{clk}$ input line by 0.3 ns w.r.t. the clk input line. As a result, a 0–0 and 1–1 overlap occurs between clk and $\overline{clk}$, which imposes a hold time constraint on the data line for a reliable transfer of data. Also, due to the overlap between clk and $\overline{clk}$, for some values of τ_h, the output reverts to the wrong logic value even after reaching 80% of the correct logic value, as shown in Fig. 16. To ensure that we are not capturing false transitions, we set the setup/hold criterion to 90% of the final output (as opposed to 50%) to calculate clock-to-q delays for this register. Hence, for a high (2.5 V) to low (0 V) transition in the data input line, we set $r = 0.25$ V in (4). The simulated value of t_c (i.e., the time at which the output reaches 90% of its final value for large setup and hold skews) was found to be equal to 12.055 ns, and therefore, t_f (i.e., the time corresponding to a 10% increase in the clock-to-q delay) was set to 12.155 ns in (4). The constant clock-to-q delay curve thus obtained by running EN curve tracing on (4) is shown in Fig. 17(a).

As in the previous example, we also plot the curve obtained by intersecting a plane at a height of 0.25 V with the output surface generated at time t_f in Fig. 17(b). The curve shown in Fig. 17(a) is also plotted on top of the intersection of the surface and the plane in Fig. 17(b). The close match is evident, again validating the correctness of EN curve tracing. Once again, EN curve tracing results in a speedup of 10.5$\times$ over brute-force output surface generation (for 17 points on the constant clock-to-q delay curve).

C. Transmission-Gate-Based Master–Slave Register

For further validation, we try to find the constant clock-to-q delay curve for the static transmission-gate-based master–slave register (Fig. 5). The clock $u_c(t)$ waveform is the same as it was in the previous two registers. This register exhibits no hold time constraint if there is no overlap between clk and $\overline{clk}$ inputs. To enforce the hold time constraint for this register, we delay the $\overline{clk}$ input line by 0.3 ns w.r.t. the clk input line. For a high (2.5 V) to low (0 V) transition in the data input line, we set $r = 0.15$ V (94% of the final output) in (4) to calculate the clock-to-q delay for this register. For very large setup and hold skews, the output reaches 0.15 V at $t_c = 12.591$ ns; hence, t_f was set equal to 12.745 ns, representing the time corresponding to a 10% increase in the clock-to-q delay. The constant clock-to-q

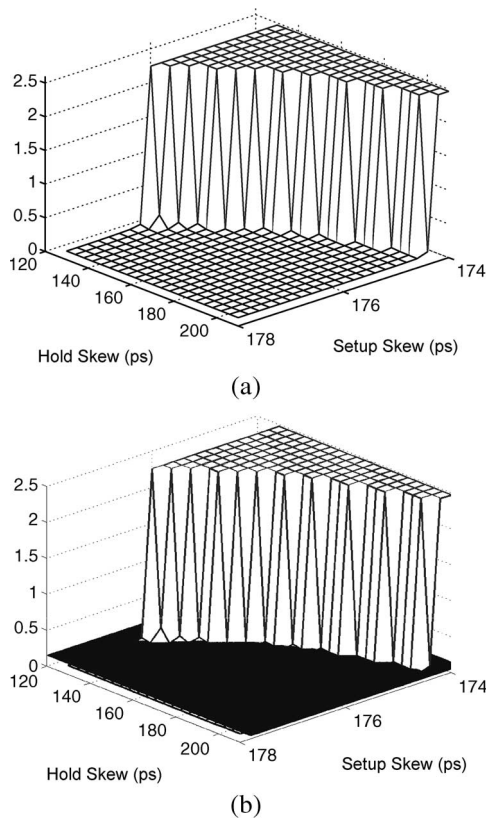

Fig. 18. (a) Q output surface as a function of independent setup and hold skews. (b) Intersection of a plane at a height of 0.15 V with the Q output surface.

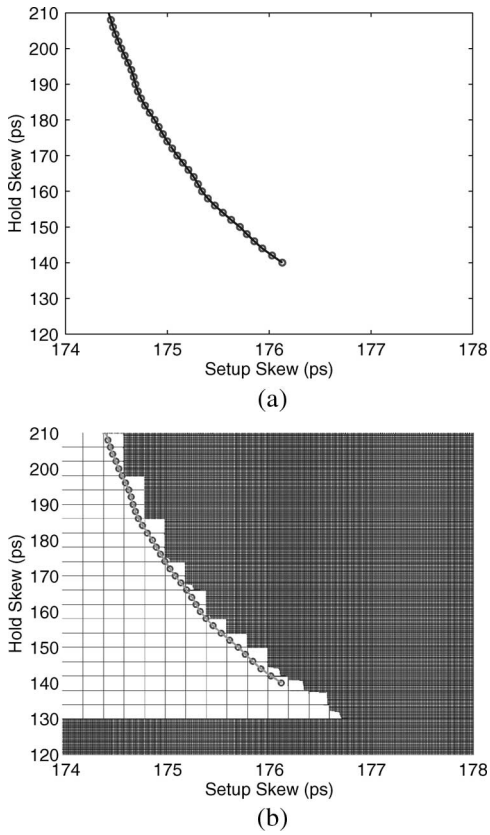

Fig. 19. (a) Contour for a 10% increase in the constant clock-to-q delay, as obtained by EN curve tracing. (b) EN curve overlaid on the intersection curve of the plane and the output surface (top view).

delay curve obtained by the EN method is shown in Fig. 19(a). This curve obtained by the EN method displays very high sensitivity w.r.t. the setup skew, as compared to the hold skew. To validate the curve, we plot the Q output surface [Fig. 18(a)] for this register at time t_f. The Q output surface in Fig. 18(a) is not very smooth and sharply changes its shape for some combinations of setup and hold skews. It can be noted from Fig. 18(a) that the Q output stays either at high (almost at 2.5 V) or at some extremely low values (near 0 V). Hence, the constant clock-to-q delay curve that we are computing using the EN method, with the criterion of the Q output at 0.15 V, actually tries to approximately match the curve that separates the extremely high and low values of the Q output surface. Therefore, to accurately trace the constant clock-to-q curve using the EN method, we need to take very small Euler steps due to the sharp transition in the Q output surface; moreover, to retain accuracy, every Newton step requires five to six iterations to converge.

The intersection curve of the Q output surface and a plane at a height of 0.15 V is shown in Fig. 18(b), and its top view is shown in Fig. 19(b). We have also overlaid the curve shown in Fig. 19(a) on top of the intersection of the surface and the plane in Fig. 19(b). As can be seen, the curve obtained by the EN method matches the intersection curve of the Q surface and the plane, even for this highly sensitive non-smooth surface. However, for this static register, the EN method is not very efficient as compared to the brute-force technique, as every Newton step took five to six iterations and very small Euler steps were needed to trace the curve for avoiding nonconvergence. Therefore, EN requires a large number of transient simulations for this particular register and gives no speedup over the brute-force method. In this case, EN does not result in any appreciable speedups over the brute-force method.

VIII. Conclusion and Future Directions

We have presented a new method that directly solves for latch *independent* setup and hold times by expressing them as solutions to a scalar nonlinear equation and solving this equation by exploiting the quadratic convergence properties of NR. We have also presented a novel method that directly solves for *interdependent* setup/hold time contours via EN curve tracing on a state-transition equation formulation. Our methods are generally applicable to any kind of latch or register. We have validated the methods on TSPC and C^2MOS register structures, demonstrating speedups of 2.5–10$\times$ for independent characterization and 10–12$\times$ for interdependent characterization over prior surface generation/intersection methods. We have also characterized a challenging transmission gate register featuring a very sharp output Q behavior and verified the correctness of the results of the new interdependent characterization method, although with no appreciable speedup in this case. We believe that the proposed interdependent characterization technique can be a crucial enabler for pessimism reduction techniques in timing analysis flows that exploit interchangeable pairs of setup/hold times [1].

APPENDIX A
COMPUTATION OF $dh(\tau)/d\tau$ FOR SOLVING FOR INDEPENDENT SETUP/HOLD TIMES VIA NR

To compute $dh(\tau)/d\tau$, we need to evaluate $(d/d\tau)\vec{\phi}(t_f; \vec{x}_0, 0, \tau)$. We next develop the procedure to do this.

First, we write out (2) with all dependences on τ explicitly shown for clarity (Note here that τ represents τ_s, and the equation will be independent of τ_h as it has been set to a fixed value.). Therefore

$$\frac{d}{dt}\vec{q}(\vec{x}(t,\tau)) + \vec{f}(\vec{x}(t,\tau)) + \vec{b}_c u_c(t) + \vec{b}_d u_d(t,\tau) = 0. \quad (21)$$

Next, noting that $d\vec{\phi}/d\tau$ is simply $d\vec{x}(t,\tau)/d\tau$, we differentiate the entire equation w.r.t. τ, interchanging the order of differentiation w.r.t. t and τ in the first term, i.e.,

$$0 = \frac{d}{dt}\left[\frac{d\vec{q}(t,\tau)}{d\vec{x}}\frac{d\vec{x}}{d\tau}\right] + \frac{d\vec{f}(t,\tau)}{d\vec{x}}\frac{d\vec{x}}{d\tau} + \vec{b}_d u_d'(t,\tau). \quad (22)$$

Since we want to evaluate $dh(\tau)/d\tau$ at any given value of τ, e.g., at τ^*, we can define the following terms for ease of understanding:

$$C^\dagger(t) = \left.\frac{d\vec{q}(t,\tau)}{d\vec{x}}\right|_{\tau=\tau^*}, G^\dagger(t) = \left.\frac{d\vec{f}(t,\tau)}{d\vec{x}}\right|_{\tau=\tau^*},$$
$$\text{and } \vec{m}^\dagger(t) = \left.\frac{d\vec{x}(t,\tau)}{d\tau}\right|_{\tau=\tau^*}. \quad (23)$$

Rewriting (22) for $\tau = \tau^*$, we obtain

$$\frac{d}{dt}\left(C^\dagger(t)\vec{m}^\dagger(t)\right) + G^\dagger(t)\vec{m}^\dagger(t) + \vec{b}_d u_d'(t,\tau^*) = 0. \quad (24)$$

The solution of (24) can easily be obtained using any integration method (e.g., BE and TRAP) [8], [9]. We write the solution of (24) using BE as

$$\frac{C^\dagger(t_i)\vec{m}^\dagger(t_i) - C^\dagger(t_{i-1})\vec{m}^\dagger(t_{i-1})}{t_i - t_{i-1}} + G^\dagger(t_i)\vec{m}^\dagger(t_i) + \vec{b}_d u_d'(t_i,\tau^*) = 0 \quad (25)$$

which can be simplified as

$$\vec{m}_i^\dagger = \left(\frac{C_i^\dagger}{\Delta t} + G_i^\dagger\right)^{-1}\left(\frac{C_{i-1}^\dagger}{\Delta t}\vec{m}_{i-1}^\dagger - \vec{b}_d u_d'(t_i,\tau^*)\right). \quad (26)$$

The subscript i denotes the fact that the corresponding quantity has been evaluated at $t = t_i$; $\Delta t = t_i - t_{i-1}$ is the time step used in the integration.

We need to know the value of $\vec{m}_0^\dagger$ to start the integration process, and it turns out that we can safely take $\vec{m}_0^\dagger$ to be equal to $\vec{0}$. The reason for this choice is that $\vec{x}_0 = \vec{x}(t = t_0)$ will not change for any particular value of τ, thus enabling $\vec{m}_0$ to take the value $\vec{0}$.

Evaluating (26) from $t = t_1$ to $t = t_f$ (i.e., $i \in 1, 2, \ldots, f$) will give $\vec{m}_f^\dagger = d\vec{\phi}_\tau/d\tau|_{t=t_f,\tau=\tau^*}$.

Finally, we can obtain the scalar

$$\left.\frac{dh(\tau)}{d\tau}\right|_{t=t_f,\tau=\tau^*} = \vec{c}^T \left.\frac{d\vec{\phi}_\tau}{d\tau}\right|_{t=t_f,\tau=\tau^*}. \quad (27)$$

APPENDIX B
COMPUTATION OF $[dh(\vec{\tau})/d\vec{\tau}]$ FOR SOLVING FOR INTERDEPENDENT SETUP/HOLD TIMES VIA MPNR

To compute $[dh(\vec{\tau})/d\vec{\tau}]$, we need to evaluate $[(d/d\tau)\vec{\phi}(t_f; \vec{x}_0, 0, \tau_s, \tau_h)]$, which is achieved as follows.

Next, noting that $[d\vec{\phi}/d\vec{\tau}] = [(d\vec{x}(t,\tau_s,\tau_h)/d\tau_s), (d\vec{x}(t,\tau_s,\tau_h)/d\tau_h]$, we first evaluate $d\vec{x}(t,\tau_s,\tau_h)/d\tau_s$; computation of $d\vec{x}(t,\tau_s,\tau_h)/d\tau_h$ follows a similar procedure.

To compute $d\vec{x}(t,\tau_s,\tau_h)/d\tau_s$, we differentiate (2) w.r.t. τ_s and interchange the order of differentiation w.r.t t and τ_s in the first term, thus obtaining

$$0 = \frac{d}{dt}\left[\frac{d\vec{q}(t,\tau_s,\tau_h)}{d\vec{x}}\frac{d\vec{x}}{d\tau_s}\right] + \frac{d\vec{f}(t,\tau_s,\tau_h)}{d\vec{x}}\frac{d\vec{x}}{d\tau_s} + \vec{b}_d z_s(t,\tau_s,\tau_h). \quad (28)$$

In (28), $(d/d\tau_s)u_d(t,\tau_s,\tau_h)$ is denoted by $z_s(t,\tau_s,\tau_h)$. Since we want to evaluate $dh(\vec{\tau})/d\tau_s$ at any given value of (τ_s,τ_h) (e.g., at (τ_s^*,τ_h^*)), we define the following terms for notational convenience:

$$C^\dagger(t) = \left.\frac{d\vec{q}(t,\tau_s,\tau_h)}{d\vec{x}}\right|_{(\tau_s^*,\tau_h^*)}, G^\dagger(t) = \left.\frac{d\vec{f}(t,\tau_s,\tau_h)}{d\vec{x}}\right|_{(\tau_s^*,\tau_h^*)},$$
$$\vec{m}_s^\dagger(t) = \left.\frac{d\vec{x}(t,\tau_s,\tau_h)}{d\tau_s}\right|_{(\tau_s^*,\tau_h^*)}. \quad (29)$$

Rewriting (28) for $(\tau_s,\tau_h) = (\tau_s^*,\tau_h^*)$, we obtain

$$\frac{d}{dt}\left(C^\dagger(t)\vec{m}_s^\dagger(t)\right) + G^\dagger(t)\vec{m}_s^\dagger(t) + \vec{b}_d z_s(t,\tau_s^*,\tau_h^*) = 0. \quad (30)$$

Equation (30) can be discretized using any integration method (e.g., BE and TRAP) [8], [9]. The discretization of (30) using BE, for example, is

$$\frac{C^\dagger(t_i)\vec{m}_s^\dagger(t_i) - C^\dagger(t_{i-1})\vec{m}_s^\dagger(t_{i-1})}{t_i - t_{i-1}} + G^\dagger(t_i)\vec{m}^\dagger(t_i) + \vec{b}_d z_s(t_i,\tau_s^*,\tau_h^*) = 0 \quad (31)$$

which can be simplified to

$$\vec{m}_{s_i}^\dagger = \left(\frac{C_i^\dagger}{\Delta t} + G_i^\dagger\right)^{-1}\left(\frac{C_{i-1}^\dagger}{\Delta t}\vec{m}_{s_{(i-1)}}^\dagger - \vec{b}_d z_s(t_i,\tau_s^*,\tau_h^*)\right). \quad (32)$$

Note that the subscript i denotes the fact that the corresponding quantity has been evaluated at $t = t_i$; $\Delta t = t_i - t_{i-1}$ is the time step used in the integration.

To start the integration process, we set $\vec{m}_{s0}^{\dagger}$ to $\vec{0}$ (the reason for this choice is that $\vec{x}_0 = \vec{x}(t = t_0)$ does not change with τ_s or τ_h).

Evaluating (32) from $t = t_1$ to $t = t_f$ (i.e., $i \in 1, 2, \ldots, f$), we obtain $\vec{m}_{sf}^{\dagger} = (d\vec{\phi}_\tau / d\tau_s)|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)}$. From these quantities, we obtain the scalar

$$\left.\frac{dh(\vec{\tau})}{d\tau_s}\right|_{(t,\tau_s,\tau_h)=\left(t_f,\tau_s^*,\tau_h^*\right)} = \vec{c}^T \left.\frac{d\vec{\phi}_\tau}{d\tau_s}\right|_{(t,\tau_s,\tau_h)=\left(t_f,\tau_s^*,\tau_h^*\right)}. \quad (33)$$

ACKNOWLEDGMENT

The authors would like to thank the following: C. Amin, C. Kashyap, and K. Killpack (Intel Strategic CAD Laboratories) for valuable discussions on the latch setup/hold characterization problem and the STA/design implications of setup/hold time tradeoffs; A. Efrati (Intel Haifa) and S. Nassif (IBM Austin Research Laboratory) for discussions related to the aforementioned problem; and the Digital Technology Center and the Supercomputing Institute, University of Minnesota, for providing computational and infrastructural resources.

REFERENCES

[1] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E. G. Friedman, "Pessimism reduction in static timing analysis using interdependent setup and hold times," in *Proc. 7th Int. Symp. Quality Electron. Des.*, Mar. 2006, pp. 159–164.

[2] W. Roethig, "Library characterization and modeling for 130 nm and 90 nm SoC design," in *Proc. IEEE Int. SOC Conf.*, Sep. 2003, pp. 383–386.

[3] D. Patel, "Charms: Characterization and modeling system for accurate delay prediction of ASIC designs," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1990, pp. 9.5.1–9.5.6.

[4] R. W. Phelps, "Advanced library characterization for high-performance ASIC," in *Proc. IEEE Cust. Int. ASIC Conf.*, Sep. 1991, pp. 15-3.1–15-3.4.

[5] T. J. Aprille and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proc. IEEE*, vol. 60, no. 1, pp. 108–114, Jan. 1972.

[6] S. Skelboe, "Computation of the periodic steady-state response of nonlinear networks by extrapolation methods," *IEEE Trans. Circuits Syst.*, vol. CAS-27, no. 3, pp. 161–175, Mar. 1980.

[7] R. Telichevesky, K. Kundert, and J. White, "Efficient AC and noise analysis of two-tone RF circuits," in *Proc. IEEE DAC*, 1996, pp. 292–297.

[8] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, ser. Prentice-Hall Series in Automatic Computation. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[9] L. O. Chua and P. M. Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computation Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes—The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1989.

[11] U. Choudhury, "Sensitivity computation in SPICE3," M.S. thesis, EECS Dept., Elect. Res. Lab., Univ. California Berkeley, Berkeley, CA, Dec. 1988.

[12] D. Hocevar, P. Yang, T. Trick, and B. Epler, "Transient sensitivity computation for MOSFET circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-4, no. 4, pp. 609–620, Oct. 1985.

[13] E. L. Allgower and K. Georg, *Numerical Continuation Methods*. New York: Springer-Verlag, 1990.

[14] K. Yamamura and M. Kiyoi, "A piecewise-linear homotopy method with the use of the Newton homotopy and polyhedral subdivision," *Trans. IEICE*, vol. 73, no. 1, pp. 140–148, 1990.

[15] S. Srivastava and J. Roychowdhury, "Interdependent latch setup/hold time characterization via Euler–Newton curve tracing on state-transition equations," in *Proc. IEEE DAC*, Jun. 2007, pp. 136–141.

[16] J. M. Rabaey, *Digital Integrated Circuits*. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[17] S. Srivastava and J. Roychowdhury, "Rapid and accurate latch characterization via direct Newton solution of setup/hold times," in *Proc. IEEE DATE Conf.*, Apr. 2007, pp. 1–6.

[18] T. L. Quarles, *SPICE 3B.1 User's Guide*. Berkeley, CA: EECS Ind. Liaison Program, Univ. California, Berkeley, Apr. 1987.

[19] T. L. Quarles, *SPICE 3C.1 User's Guide*. Berkeley, CA: EECS Ind. Liaison Program, Univ. California, Berkeley, Apr. 1989.

Shweta Srivastava received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 2002, and the M.S. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2007.

From 2003 to 2005, she was an Associate Design Engineer with STmicroelectronics Pvt. Ltd., Noida, India. She is currently a Research and Development Engineer with Synopsys, Inc., Mountain View, CA. Her research interests include design and simulation of mixed-signal circuits, latch characterization, macromodeling, and fast simulation of nanotechnological and biochemical systems.

Jaijeet Roychowdhury received the B.S. degree from the Indian Institute of Technology, Kanpur, India, in 1987, and the Ph.D. degree from the University of California, Berkeley, in 1993, both in electrical engineering.

From 1993 to 1995, he was with the Computer-Aided Design Laboratory, AT&T Bell Laboratories, Allentown, PA. From 1995 to 2000, he was with the Communication Sciences Research Division, Bell Laboratories, Murray Hill, NJ. From 2000 to 2001, he was with CeLight Inc. (an optical networking startup), Silver Spring, MD. Since 2001, he has been with the Department of Electrical and Computer Engineering and the Digital Technology Center, University of Minnesota, Minneapolis. Over the years, he has authored or coauthored seven best or distinguished papers at ASP-DAC, DAC, and ICCAD. He is the holder of ten patents. His research interests include the analysis, simulation, and design of electronic, biological, and mixed-domain systems.

Dr. Roychowdhury was an IEEE Circuits and Systems Society Distinguished Lecturer from 2003 to 2005, has served as Program Chair of IEEE's CANDE and BMAS workshops in 2005, and is currently the Technical Program Chair of ICCAD. He serves, or has served on, the Technical Program Committees of DAC, ICCAD, DATE, ASP-DAC, ISQED, IDV, BMAS, and IASTED, on the Executive Committee of ICCAD, on the Nominations and Appointments Committee of CEDA, and as an officer of CANDE. He was cited for Extraordinary Achievement by Bell Laboratories in 1996 (for work on practically effective numerical continuation methods).