Interdependent Latch Setup/Hold Time Characterization via Euler-Newton Curve Tracing on State-Transition Equations

Shweta Srivastava, Jaijeet Roychowdhury
Department of Electrical and Computer Engineering, University of Minnesota
Email: {shwetas,jr}@umn.edu

Abstract

Interdependent characterization of latch setup/hold times is a core component of techniques for pessimism reduction in timing analysis [1]. We present an efficient and novel method for such characterization, by formulating the interdependent setup-hold time problem as an *underdetermined* nonlinear equation $h(\tau_s, \tau_h) = 0$. We solve this equation, which is derived from the latch's state-transition function, numerically using a Moore-Penrose Newton method. Further, 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 the method on TSPC and C^{2}MOS registers, obtaining speedups of more than 20$\times$ over prior approaches while achieving superior accuracy. This speedup increases linearly with the precision with which curve tracing is desired. In view of the importance and large computational expense of latch characterization in industry today, the new technique represents a significant enabling technology for dramatically speeding up industrial timing closure flows.

I. Introduction

Accurately characterizing setup and hold times of latches and registers is crucially important for static and dynamic timing analysis of digital circuits [1]–[3]. With aggressive technology scaling, devices have been becoming faster, but also much more non-ideal; therefore, approximate characterization, typically based on idealizations, are becoming markedly less valid, especially for circuits with memory such as latches and registers. At the same time, faster speeds and smaller design margins make it all the more important to characterize with high precision, to achieve realistic timing calculations devoid of unnecessary optimism or pessimism[1] [1], [4]. As a result, full SPICE-level transient analysis of latch/register circuits, using detailed device models, 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.

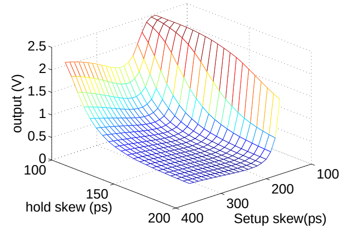
In general practice today, setup and hold times are determined independently, *i.e.* it is assumed that the two quantities are not correlated with each other. However, the fact that setup and hold times are interdependent is in fact well known (*e.g.*, [1]); *i.e.*, multiple pairs of setup and hold times are possible that result in the same clock-to-q delay[2]. Flexibility in trading off setup vs hold time is important for reducing violations in static timing analysis [1] without sacrificing performance.

The prevalent technique today 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 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[3], 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%.

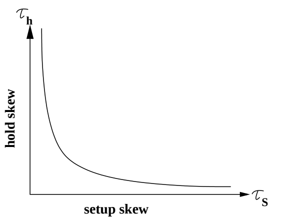
[1]Optimism in setup/hold times can cause circuit failure, while pessimism results in inferior performance.

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

One such surface for a register output Q vs 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 times pairs of interest for timing analysis.



(a) Q output surface.



(b) Constant clock-to-q delay curve.

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

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 post-processing output surfaces like 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.

In this paper, we adapt ideas from mixed-signal/RF simulation and from homotopy/numerical continuation [5]–[9] to devise a new technique for finding interdependent setup/hold time contours directly, 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 [10], [11] of the differential equations describing the latch or register, as described in more detail later. We then use a modified Moore-Penrose pseudo-inverse based Newton-Raphson (MPNR) method [8], [12] to solve the equation $h(\tau_s, \tau_h) = 0$ numerically for *one pair* of interdependent setup/hold times. Once this point is found, our algorithm proceeds to determine other points on the constant clock-to-q contour efficiently by an Euler-Newton curve tracing procedure [8].

Euler-Newton curve tracing operates by using information from a given solution point on the clock-to-q delay curve to solve for a neighboring point on the curve efficiently. 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[4] 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 of irrelevant points on the surface is totally avoided. As a result, the number of

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

[4]*i.e.*, 2-3 MPNR iterations is typical.

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, Euler-Newton curve tracing provides speedups of about n times over output surface generation, where n, 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 (TPSC) and C^{2}MOS registers, we obtain speedups of about 26× for $n = 40$ points on the curve. Additionally, the points obtained on the curve by Euler-Newton method are "exact" (*i.e.*, refined to any prescribed accuracy by MPNR), while the brute-force technique uses interpolation at the postprocessing stage to extract the contour from the output surface.

The remainder of the paper is organized as follows. We first provide some background on the register setup/hold time problem in Section II. Section III is divided into five subsections. Section III*A* formulates the interdependent setup/hold problem as an equation $h(\tau_s, \tau_h) = 0$; Section III*B* provides a brief discussion of the special case of solving for τ_s and τ_h independently of each other; Section III*C* develops the procedure for solving $h(\tau_s, \tau_h) = 0$ by Moore-Penrose pseudo-inverse based Newton-Raphson, focusing on a single point on the curve; Section III*D* outlines the Euler-Newton method for tracing the solution curve of $h(\tau_s, \tau_h) = 0$; finally, Section III*E* puts together the complete Euler-Newton curve-tracing algorithm for interdependent setup/hold times using a pseudo-code description. In Section IV, we validate the new technique on practical latch/register circuits and compare against brute-force output surface generation, confirming that Euler-Newton successfully traces the constant clock-to-q delay curve accurately and with large speedups.

II. Terminology and Brief Background

Latches and registers are basic building blocks in synchronous circuit design; in essence, they are circuits where a clock edge is used to sample and store a logic value on a data line [13]. The *setup time* is the time before the active edge of the clock that the input data line must be valid for reliable latching. Similarly, the *hold time* represents the time that the data input must be held stable after the active clock edge. Active clock edge of the register is either low-to-high or high-to-low transition edge at which data transfer happens.

Clock-to-Q delay is a common term 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. Setup and hold skew are denoted by τ_s and τ_h, respectively, in Fig. 2.

As already noted, setup and hold time are not independent quantities, but depend strongly on each other. Typically, 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 Interdependent Register Setup and Hold Time Characterization

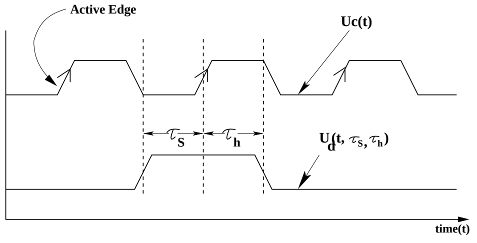


Fig. 2. Clock/data waveforms.

In this section, we first develop a formulation that expresses interdependent setup/hold times as an underdetermined scalar equation. We show how to solve the equation using a Moore-Penrose pseudo-inverse Newton procedure. We then develop a Euler-Newton curve tracing procedure for this equation.

A. Interdependent Setup/Hold Time Formulation

Any nonlinear circuit or system can be represented by the following vector differential algebraic equation [11]:

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

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

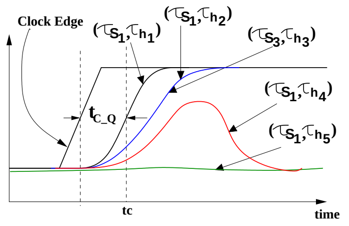
A typical register, consisting of many transistors, has a clock line and one or more data lines as input. It is a nonlinear circuit system and can be represented by (1).

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 represents 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 bring out its dependence on τ_s and τ_h explicitly.

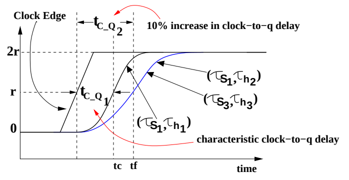
Once the clock and data inputs (refer to Fig. 2) are separated as above, the differential equations of the register can be written as follows:

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

In order 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, given by $\vec{c}^T\vec{x}$. Here, $\vec{c}$ will typically be a unit vector which 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}, 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}) point to the same output waveform, *i.e.* they 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 and approaches the *characteristic clock-to-q delay* of the register.



(a) Output for $\tau_{h1} > \tau_{h2} > \tau_{h3} > \tau_{h4} > \tau_{h5}$.



(b) Clock to Q delay.

Fig. 3. Behavior of output waveform for different setup skews.

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 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 depicted graphically in Fig. 3(b) — t_{C-Q_1} represents the characteristic clock-to-q delay and t_{C-Q_2} a 10% increase in t_{C-Q_1}. Therefore, setup-hold skew pairs that lead to 10% degradation in clock-to-q delay in the second waveform qualify as the 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 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

$$
\begin{aligned}
&\vec{c}^T\vec{\phi}(t_f;\vec{x}_0,0,\tau_s,\tau_h)-r=0,\\
&\text{or } \vec{c}^T\vec{\phi}_\tau(\tau_s,\tau_h)-r=0, \qquad (3)\\
&\text{where } \vec{\phi}_\tau(\tau_s,\tau_h)\equiv\vec{\phi}(t_f;\vec{x}_0,0,\tau_s,\tau_h).
\end{aligned}
$$

Hence, the nonlinear equation we need to solve to obtain (τ_s,τ_h) is

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

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

B. Prior work: independent solution of setup or hold time

In this subsection, we mention briefly 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, since the setup time becomes independent for very large value of hold skew, (4) can be written as follows for very large τ_h:

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

(5) is a scalar equation with one scalar unknown τ_s, hence can be solved using the standard Newton-Raphson method [12]. In prior work that will appear elsewhere, we have shown how to solve (5) using NR, achieving speedups of $4\sim10\times$ over the current practice of binary search. We do not recount the details of this procedure here since it is not central to the curve tracing procedure for interdependent setup/hold time that is the main contribution of this work.

C. Solving for interdependent setup/hold times via Moore-Penrose based Newton-Raphson

Since (4) is an underdetermined scalar nonlinear equation with two unknowns τ_s and τ_h, Newton-Raphson, the method of choice for numerical solution of "square" nonlinear systems, cannot be directly applied. However, a modification, based on the application of the Moore-Penrose pseudo-inverse [8] during the linear solution step, can be applied instead, as described in this subsection. Intuitively,

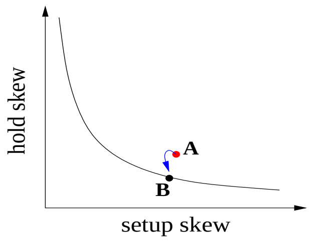


Fig. 4. Convergence of a point via Newton-Raphson on a constant clock-to-q delay curve.

Moore-Penrose Newton-Raphson (MPNR) starts with an initial guess of (τ_{s0},τ_{h0}) and converges to a solution $({\tau_s}^c,{\tau_h}^c)$ of (4), which lies on the constant clock-to-q delay curve, as shown in Fig. 4. *A* denotes the initial guess, while *B* denotes the point on the solution curve Fig. 4 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*.

In order 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 $\left[\frac{dh(\vec{\tau})}{d\vec{\tau}}\right]=\left[\frac{dh(\vec{\tau})}{d\tau_s},\frac{dh(\vec{\tau})}{d\tau_h}\right]$, a 1x2 matrix, and 3) compute the Moore-Penrose pseudo inverse of $\left[\frac{dh(\vec{\tau})}{d\vec{\tau}}\right]$.

$h(\tau_s,\tau_h)$ is evaluated simply by running a transient simulation with the given (τ_s,τ_h) and then evaluating (4). To compute $\left[\frac{dh(\vec{\tau})}{d\vec{\tau}}\right]$, we need to evaluate $\left[\frac{d}{d\tau}\vec{\phi}(t_f;\vec{x}_0,0,\tau_s,\tau_h)\right]$, which is achieved as follows.
.

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

First, we write out (2) with all dependencies on τ_s and τ_h explicitly shown for clarity:

$$\frac{d}{dt}\vec{q}(\vec{x}(t,\tau_s,\tau_h))+\vec{f}(\vec{x}(t,\tau_s,\tau_h))+\vec{b}_c u_c(t)+\vec{b}_d u_f(t,\tau_s,\tau_h)=0. \qquad (6)$$

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

To compute $\frac{d\vec{x}(t,\tau_s,\tau_h)}{d\tau_s}$, we differentiate (6) with respect to τ_s and interchange the order of differentiation w.r.t t and τ_s in the first term, to obtain:

$$
\begin{aligned}
0&=\frac{d}{d\tau_s}\left[\frac{d}{dt}\vec{q}(\vec{x}(t,\tau_s,\tau_h))+\vec{f}(\vec{x}(t,\tau_s,\tau_h))+\vec{b}_c u_c(t)+\vec{b}_d u_d(t,\tau_s,\tau_h)\right]\\
&=\frac{d}{dt}\left[\frac{d}{d\tau_s}\vec{q}(\vec{x}(t,\tau_s,\tau_h))\right]+\frac{d}{d\tau_s}\left[\vec{f}(\vec{x}(t,\tau_s,\tau_h))\right]+\vec{b}_d\frac{d}{d\tau_s}u_d(t,\tau_s,\tau_h)\\
&=\frac{d}{dt}\left[\frac{d\vec{q}(t,\tau_s,\tau_h)}{d\vec{x}}\frac{d\vec{x}}{d\tau_s}\right]+\frac{d\vec{f}(t,\tau_s,\tau_h)}{d\vec{x}}\frac{d\vec{x}}{d\tau_s}+\vec{b}_d z_s(t,\tau_s,\tau_h).
\end{aligned}
\qquad (7)
$$

In (7), $\frac{d}{d\tau_s}u_d(t,\tau_s,\tau_h)$ is denoted by $z_s(t,\tau_s,\tau_h)$. Since we want to evaluate $\frac{dh(\vec{\tau})}{d\tau_s}$ at any given value of (τ_s,τ_h) (*e.g.*, at (τ_s^*,τ_h^*)), we define the following terms for notational convenience:

$$
\begin{aligned}
C^\dagger(t)&=\left.\frac{d\vec{q}(t,\tau_s,\tau_h)}{d\vec{x}}\right|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)},\\
G^\dagger(t)&=\left.\frac{d\vec{f}(t,\tau_s,\tau_h)}{d\vec{x}}\right|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)},\\
\text{and}\quad \vec{m}_s{}^\dagger(t)&=\left.\frac{d\vec{x}(t,\tau_s,\tau_h)}{d\tau_s}\right|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)}.
\end{aligned}
\qquad (8)
$$

Rewriting (7) for $(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)$, we obtain

$$\frac{d}{dt}\left(C^\dagger(t)\vec{m}_s{}^\dagger(t)\right)+G^\dagger(t)\vec{m}_s{}^\dagger(t)+\vec{b}_d z_s(t,\tau_s^*,\tau_h^*)=0. \qquad (9)$$

(9) can be discretized using any integration method (*e.g.* BE, TRAP *etc.*) [10], [11]. The discretization of (9) using BE (for example) is

$$
\begin{aligned}
&\frac{C^\dagger(t_i)\vec{m}_s{}^\dagger(t_i)-C^\dagger(t_{i-1})\vec{m}_s{}^\dagger(t_{i-1})}{t_i-t_{i-1}}\\
&\quad+G^\dagger(t_i)\vec{m}^\dagger(t_i)+\vec{b}_d z_s(t_i,\tau_s^*,\tau_h^*)=0,
\end{aligned}
\qquad (10)
$$

which can be simplified to

$$\vec{m}_{s}{}_i^\dagger=\left(\frac{C_i^\dagger}{\Delta t}+G_i^\dagger\right)^{-1}\left(\frac{C_{i-1}^\dagger}{\Delta t}\vec{m}_{s}{}_{(i-1)}^\dagger-\vec{b}_d z_s(t_i,\tau_s^*,\tau_h^*)\right). \qquad (11)$$

Note: 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.

Also define:

$$
\begin{aligned}
z_h(t,\tau_s,\tau_h)&=\frac{d}{d\tau_h}u_d(t,\tau_s,\tau_h)\\
\text{and}\quad \vec{m}_h{}^\dagger(t)&=\left.\frac{d\vec{x}(t,\tau_s,\tau_h)}{d\tau_h}\right|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)};
\end{aligned}
\qquad (12)
$$

then, similar to (11), we have:

$$\vec{m}_{h}{}_i^\dagger=\left(\frac{C_i^\dagger}{\Delta t}+G_i^\dagger\right)^{-1}\left(\frac{C_{i-1}^\dagger}{\Delta t}\vec{m}_{h}{}_{(i-1)}^\dagger-\vec{b}_d z_h(t_i,\tau_s^*,\tau_h^*)\right). \qquad (13)$$

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

Evaluating (11) and (13) from $t=t_1$ to $t=t_f$ (*i.e.* $i\in 1,2,\ldots,f$), we obtain $\vec{m}_{s}{}_f^\dagger=\left.\frac{d\vec{\phi}_\tau}{d\tau_s}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)}$ and $\vec{m}_{h}{}_f^\dagger=$

$\left.\frac{d\vec{\phi}_\tau}{d\tau_h}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)}$. From these quantities, we obtain the scalar

$$\begin{aligned}\left.\frac{dh(\vec{\tau})}{d\tau_s}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)} &= \vec{c}^T \left.\frac{d\vec{\phi}_\tau}{d\tau_s}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)}, \\ \left.\frac{dh(\vec{\tau})}{d\tau_h}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)} &= \vec{c}^T \left.\frac{d\vec{\phi}_\tau}{d\tau_h}\right|_{(t,\tau_s,\tau_h)=(t_f,\tau_s^*,\tau_h^*)}.\end{aligned} \tag{14}$$

Finally, denoting the matrix $\left[\frac{dh(\vec{\tau})}{d\vec{\tau}}\right]$ by $H(\vec{\tau})$, its Moore-Penrose pseudo-inverse [8] can be expressed as

$$H(\vec{\tau})^+\Big|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)} = H(\vec{\tau})^t\left(H(\vec{\tau})H(\vec{\tau})^t\right)^{-1}\Big|_{(\tau_s,\tau_h)=(\tau_s^*,\tau_h^*)}, \tag{15}$$

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

MPNR algorithm is given in detail in subsection E.

D. Constant Clock-to-Q Delay Contour Tracing by Euler-Newton

In the previous subsection, 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 skew is given. In this subsection, we outline an Euler-Newton 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 Euler-Newton curve tracing [8] method used here follows a standard predictor-corrector methodology [10], 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 Euler-Newton curve tracing procedure is depicted graphically in Fig. 5, where the blue and red arrows denote the Euler predictor steps and the MPNR corrector steps, respectively.

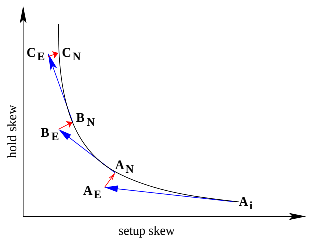


Fig. 5. Curve tracing using the Euler-Newton method.

The previous subsection 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 $({\tau_s}^c, {\tau_h}^c)$ 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 [8]:

$$T(H(\vec{\tau}))|_{\vec{\tau}=\vec{\tau}^c} = \left.\begin{pmatrix} -\frac{dh(\vec{\tau})}{d\tau_h} \\ \frac{dh(\vec{\tau})}{d\tau_s}\end{pmatrix} \frac{1}{\sqrt{\left(\frac{dh(\vec{\tau})}{d\tau_s}\right)^2 + \left(\frac{dh(\vec{\tau})}{d\tau_h}\right)^2}}\right|_{\vec{\tau}=\vec{\tau}^c}. \tag{16}$$

The rectangular matrix in (16) is simply the MPNR Jacobian matrix at the current solution point, hence is already available; since it is of size 2, the computation involved in finding the tangent vector is trivial.

E. Euler-MPNR-based curve tracing algorithm for interdependent setup/hold time characterization

We now outline the overall procedure of finding all the pairs $({\tau_s}^c, {\tau_h}^c)$ such that they satisfy (4). (4) is rewritten below for convenience:

$$h(\tau_s, \tau_h) = \vec{c}^T \vec{\phi}_\tau(\tau_s, \tau_h) - r = 0. \tag{17}$$

The above equation needs to be evaluated at time $t = t_f$. We note again that the curve obtained as the solution set of (17) 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:

1) Initialize τ_s, τ_h, $\vec{x}$, $\vec{m}_s$ and $\vec{m}_h$.
   a) $(\tau_s, \tau_h) = (\tau_{s0}, \tau_{h0})$.
      (τ_{s0}, τ_{h0}) can be chosen to be any positive value; a good choice will always be less than the time period of the clock. A good guess of (τ_{s0}, τ_{h0}) will typically approximate some previously known pair of setup and hold time of the similar kind of registers. It is necessary also to ensure that (τ_{s0}, τ_{h0}) falls in the convergence range of NR.
   b) $\vec{x}(t=0, \vec{\tau}) = \vec{x}_0(\vec{\tau})$.
      $\vec{x}_0(\vec{\tau})$ can be made identical for all values of $\vec{\tau}$, *i.e.*, equal to some $\vec{x}_0^*$, where $\vec{x}_0^*$ can assume any arbitrary value.
      A good choice of $\vec{x}_0(\vec{\tau})$ is the DC operating point for that particular value of $\vec{\tau}$; here $\vec{x}_0(\vec{\tau})$ will differ for different values of $\vec{\tau}$'s.
   c) $\vec{m}_s(t=0, \vec{\tau}) = \vec{m}_{s0}(\vec{\tau})$ and $\vec{m}_h(t=0, \vec{\tau}) = \vec{m}_{h0}(\vec{\tau})$.
      As explained previously, $\vec{m}_{s0}(\vec{\tau})$ and $\vec{m}_{s0}(\vec{\tau})$ will be initialized to $\vec{0}$ because $\vec{x}_0$ does not change for with $\vec{\tau}$.
2) Start the Euler-Newton 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 Newton-Raphson 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:
         $\vec{\tau}_{kj} = \left[\tau_{skj}, \tau_{hkj}\right]$ *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) (reproduced below).
            Here, $\vec{x}_{kji}$ denotes to 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 hold for other quantities too.

$$\frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}_c u_c(t) + \vec{b}_d u_d(t, \tau_s, \tau_h) = 0. \tag{18}$$

            (18) can be solved using any integration method like BE, TRAP *etc.* [10], [11].
         B) After having obtained $\vec{x}_{kji}$'s, compute the following:

$$C_{kji} = \left.\frac{d\vec{q}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{kji}} \text{ and } G_{kji} = \left.\frac{d\vec{f}(\vec{x})}{d\vec{x}}\right|_{\vec{x}=\vec{x}_{kji}}. \tag{19}$$

         C) Compute $(\vec{m}_s)_{kji}$ and $(\vec{m}_h)_{kji}$ using (11) and (13) as follows.

$$\begin{aligned}(\vec{m}_s)_{kji} &= \left(\frac{C_{kji}}{t_i - t_{i-1}} + G_{kji}\right)^{-1} \times \\ &\quad \left(\frac{C_{kj(i-1)}}{t_i - t_{i-1}}(\vec{m}_s)_{kj(i-1)} - \vec{b}_d z_s(t_i, \vec{\tau}_{kj})\right), \\ (\vec{m}_h)_{kji} &= \left(\frac{C_{kji}}{t_i - t_{i-1}} + G_{kji}\right)^{-1} \times \\ &\quad \left(\frac{C_{kj(i-1)}}{t_i - t_{i-1}}(\vec{m}_h)_{kj(i-1)} - \vec{b}_d z_h(t_i, \vec{\tau}_{kj})\right).\end{aligned} \tag{20}$$

         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.

$$h(\vec{\tau}_{kj}) = \vec{c}^T \vec{x}_{kj(N-1)} - r. \tag{21}$$

      iii) Check convergence of NR using reltol and abstol [14]. If NR has converged, then we have obtained the optimal value of $\vec{\tau}$ as $\vec{\tau}_{kj}$. Jump to step(b) to compute Euler step at $\vec{\tau}_{kj}$.

Also, we will denote this optimal value of $\vec{\tau}_{kj}$, which essentially lies on the curve, as $\vec{\tau}_k{}^c$ outside the NR procedure.

Otherwise calculate $\frac{dh(\vec{\tau})}{d\tau}$ defined in (14) as follows:

$$\left.\frac{dh(\vec{\tau})}{d\tau_s}\right|_{\vec{\tau}=\vec{\tau}_{kj}} = \vec{c}^T(\vec{m}_s)_{kj(N-1)}, \qquad \left.\frac{dh(\vec{\tau})}{d\tau_h}\right|_{\vec{\tau}=\vec{\tau}_{kj}} = \vec{c}^T(\vec{m}_h)_{kj(N-1)}. \tag{22}$$

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

$$\tau_{k(j+1)} = \tau_{kj} - h(\vec{\tau}_{kj})\left.\left(\frac{dh(\vec{\tau})}{d\tau}\right)^{+}\right|_{\vec{\tau}=\vec{\tau}_{kj}}, \tag{23}$$

and $\quad j = j+1$.

In (23), $\left(\frac{dh(\vec{\tau})}{d\tau}\right)^{+}$ is the Moore-Penrose pseudo-inverse of $\frac{dh(\vec{\tau})}{d\tau}$, which can be calculated using (15) as follows:

$$\left(\frac{dh(\vec{\tau})}{d\tau}\right)^{+} = \left(\frac{dh(\vec{\tau})}{d\tau}\right)^{t}\left[\frac{dh(\vec{\tau})}{d\tau}\left(\frac{dh(\vec{\tau})}{d\tau}\right)^{t}\right]^{-1} \tag{24}$$

Where $\left(\frac{dh(\vec{\tau})}{d\tau}\right)^{t}$ is the transpose of $\frac{dh(\vec{\tau})}{d\tau}$.

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

b) Compute the tangent unit vector induced by $\frac{dh(\vec{\tau})}{d\tau}$ using (16) as following for $\vec{\tau} = \vec{\tau}_k{}^c$:

$$T\left(\frac{dh(\vec{\tau})}{d\tau}\right) = \begin{pmatrix} -\frac{dh(\vec{\tau})}{d\tau_h} \\ \frac{dh(\vec{\tau})}{d\tau_s} \end{pmatrix} \frac{1}{\sqrt{\left(\frac{dh(\vec{\tau})}{d\tau_s}\right)^2 + \left(\frac{dh(\vec{\tau})}{d\tau_h}\right)^2}} \tag{25}$$

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

$$\vec{\tau}_{(k+1)0} = \vec{\tau}_k{}^c + \alpha.T\left.\left(\frac{dh(\vec{\tau})}{d\tau}\right)\right|_{\vec{\tau}=\vec{\tau}_k{}^c}, \tag{26}$$

and $\quad k = k+1$.

Where α is the step used in the direction of tangent. Go to step(a) of Euler-Newton procedure and keep repeating until the traversing is stopped.

IV. Validation using TSPC and C^{2}MOS registers

In this section, we validate the interdependent setup/hold time characterization algorithm developed in the paper using two types of registers: a TPSC register and a C^{2}MOS positive edge triggered master/slave register. We describe the validation procedure in some detail. Our results, which we compare against brute-force output surface generation, confirm that the new curve-tracing method finds constant clock-to-q contours accurately in computation linear in the number of contour points desired. We obtain speedups of about 26× over surface generation[5] for 40 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 5 digits.

A. True single-phased clocked register

The positive edge triggered true single-phased clocked register shown in Fig. 6 features positive setup and hold time constraints. The clock waveform $u_c(t)$ used in the register has a period of 10ns, with logic 0 at 0V and logic 1 at 2.5V. The clock input has an initial delay of 1ns; rise/fall times are both 0.1ns. Therefore, its active clock edges are located at 1ns, 11ns, 21ns, *etc.*. The chosen data waveform $u_d(t, \tau_s, \tau_h)$ is centered around the active clock edge which starts at

[5]Speedup numbers are obtained via apples-to-apples comparisons on an AMD Athlon64 3000+ based PC, with 512MB RAM, running Linux kernel 2.6.12. All algorithms are implemented in a MATLAB/C/C++ simulation prototyping environment.

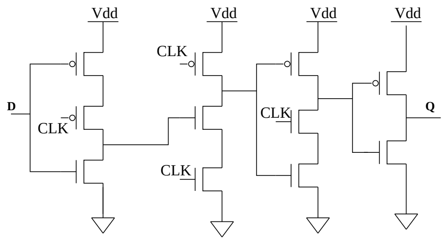
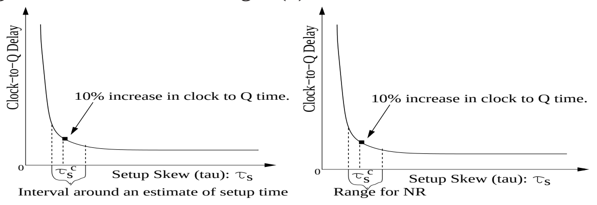


Fig. 6. Positive edge-triggered register in TSPC.

11ns. 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 1.25V (50% of its final value) at $t_c = 11.348ns$, hence the characteristic clock-to-q delay equals 298ps (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: that they lead to an increase of 10% over the characteristic clock-to-q delay. Hence the constant clock-to-q delay value for contour generation equals 327.8ps. Accordingly, we set $t_f = 11ns + \frac{rise-time}{2} + 327.8ps = 11.3778ns$ and $r = 1.25V$ 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 becomes largely independent of hold skew. We start with a setup skew interval $[\tau_{sL}, \tau_{sR}]$, where the register latches the data properly for τ_{sL}, and fails to latch data for τ_{sR}. Hence this interval will contain the setup time point $\tau_s{}^c$. We then narrow down the setup skew interval using a coarse binary search, until the interval length falls in the convergence range of NR as shown in Fig. 7(b).



(a) Setup-time characterization. (b) Convergence region for MPNR.

Fig. 7. (a) Setup time characterization by running binary search within an interval bracketing the setup time. (b) Convergence region of MPNR in an interval around the estimate of setup time.

Either τ_{sL} or τ_{sR} can be used as the initial guess τ_{s0} for MPNR solution. Then, we start the Euler-Newton process as outlined previously. MPNR typically converges very quickly (2–3 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. 8. To verify the correctness of this curve, we also

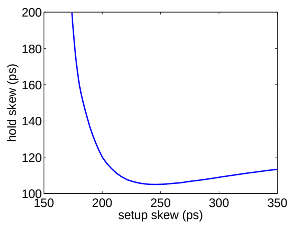


Fig. 8. Constant clock-to-q delay curve obtained by the Euler-Newton method. Curve represents the pairs of setup and hold skews which increases the characteristic clock-to-q delay by 10%.

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 $t_f = 11.778ns$ by independently varying setup and hold skews as shown in Fig. 9. A plane at a "height" of 1.25V is then drawn to obtain the intersection contour of the output surface. This intersection curve represents the set of all hold-setup skew pairs which results in 10% increase in clock-to-q delay. The top view of the curve obtained by this intersection

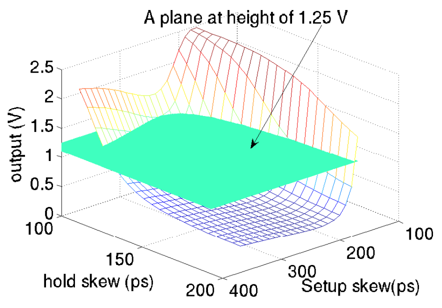


Fig. 9. Q output surface as a function of independent setup and hold skews.

procedure is shown in Fig. 10. The contour from Euler-Newton curve tracing (Fig. 8) is overlaid on the intersection of the plane and the output surface in Fig. 10. It is apparent from Fig. 10 that the curve obtained by Euler-Newton methodology exactly matches the constant clock-to-q delay curve obtained from the surface, thereby verifying the correctness of the new method.

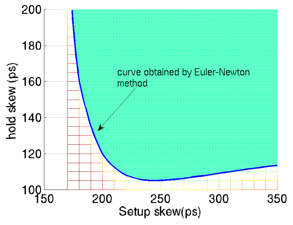
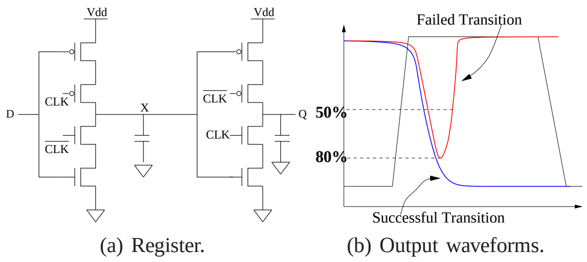


Fig. 10. Top view of intersection curve of plane and output surface and the superimposition of curve obtained by Euler-Newton method

In our implementation, Euler-Newton curve tracing took 45 minutes to trace 40 points on the contour, while brute-force output surface generation consumed 20 hours (for 40×40 simulations) to obtain 40 points on the curve; representing a speedup of about $26\times$.

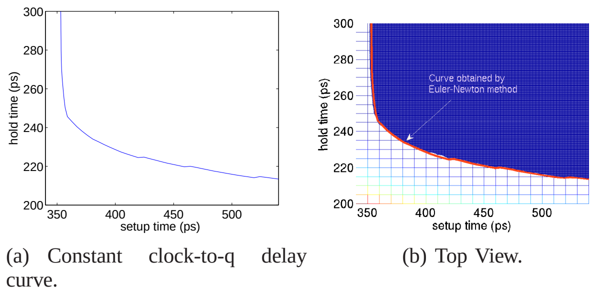
B. C^{2}MOS positive-edge triggered master-slave register



(a) Register. (b) Output waveforms.

Fig. 11. (a) C^2MOS positive-edge triggered master-slave register. (b) Output waveform fails to complete the transition even after reaching 80% of its true value.

To further validate the Euler-Newton curve-tracing method, we apply it to the C^{2}MOS register shown in Fig. 11(a). 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. 11(a) 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 $0.3ns$ 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 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 correct logic value, as shown in Fig. 11(b). 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 ($2.5V$) to low ($0V$) transition in the data input line, we set $r = 0.25V$ in (4). The simulated value of t_c (*i.e.*, the time at which output reaches 90% of its final value for large setup and hold skews) was found to be equal to $12.055ns$, and therefore t_f (*i.e.*, the time corresponding to a 10% increase in clock-to-q delay) was set to 12.155ns in (4). The constant clock-to-q delay curve thus obtained by running Euler-Newton curve tracing on (4) is shown in Fig. 12(a). As in the previous example, we also plot the curve obtained by the



(a) Constant clock-to-q delay curve. (b) Top View.

Fig. 12. (a) Contour for 10% increase in clock-to-q delay, obtained by Euler-Newton curve tracing. (b) Euler-Newton curve overlaid on the intersection curve of the plane and the output surface.

intersecting a plane at height $0.25V$ with the output surface generated at time t_f in Fig. 12(b). The curve shown in Fig. 12(a) is also plotted on top of the intersection of the surface and the plane in Fig. 12(b). The close match is evident, again validating the correctness of Euler-Newton curve tracing. Once again, Euler-Newton curve tracing results in a speedup of $26.66\times$ over brute-force output surface generation (for 40 points on the constant clock-to-q delay curve).

CONCLUSIONS

We have presented a novel method that directly solves for inter-dependent setup-hold time contours via Euler-Newton curve tracing on a state-transition equation formulation. The method is generally applicable to any kind of latch or register. We have validated the method on TSPC and C^{2}MOS register structures and demonstrated speedups of $26\times$ over prior surface generation/intersection methods. The new technique can be a crucial enabler for pessimism reduction techniques in timing analysis flows that exploit interchangeable pairs of setup/hold times [1].

REFERENCES

[1] Emre Salman, Ali Dasdan, Feroze Taraporevala, Kayhan Kucukcakar, and Eby G. Friedman. Pessimism reduction in static timing analysis using interdependent setup and hold times. *Proceedings of the 7th International Symposium on Quality Electronic Design*, page 6, March 2006.

[2] W. Roethig. Library characterization and modeling for 130 nm and 90 nm soc design. *Proceedings of the IEEE International SOC Conference*, pages 383–386, September 2003.

[3] D. Patel. Charms: Characterization and modeling system for accurate delay prediction of asic designs. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 9.5.1–9.5.6, May 1990.

[4] R. W. Phelps. Advanced library characterization for high-performance asic. *Proceedings of the IEEE Custom International ASIC conference*, pages 15–3.1 – 15–3.4, September 1991.

[5] T.J. Aprille and T.N. Trick. Steady-state analysis of nonlinear circuits with periodic inputs. *Proc. IEEE*, 60(1):108–114, January 1972.

[6] S. Skelboe. Computation of the periodic steady-state response of nonlinear networks by extrapolation methods. *IEEE Trans. Ckts. Syst.*, CAS-27(3):161–175, March 1980.

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

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

[9] K. Yamamura and M. Kiyoi. A piecewise linear homotopy method with the use of the Newton homotopy and polyhedral subdivision. *Trans.IEICE*, 73:140–148, 1990.

[10] C.W. Gear. *Numerical initial value problems in ordinary differential equations*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, N.J., 1971.

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

[12] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes – The Art of Scientific Computing*. Cambridge University Press, 1989.

[13] Jan M. Rabaey. *Digital Integrated Circuits*. Prentice-Hall, 2004.

[14] Thomas L. Quarles. *SPICE 3C.1 User's Guide*. University of California, Berkeley, EECS Industrial Liaison Program, University of California, Berkeley California, 94720, April 1989.