State Transformation in Event Driven Explicit Simulation

Tuyen V. Nguyen and Anirudh Devgan
IBM Austin Research Laboratory
Austin, TX 78758
tuyenn;devgan@austin.ibm.com

Abstract
This paper presents a general method for incorporating state transformation in event driven explicit simulation. One inherent assumption in this type of simulation algorithm is the state independence which allows the algorithm to process the states independently in an event driven manner at the transistor level. Numerical problems arise when an inappropriate state representation of the circuit, in which the states are not truly independent, is chosen. In principle, any similarity transformation of the state equation can be employed to transform the circuit into a more convenient state space for numerical solution. This paper develops a systematic scheme to derive an appropriate state transformation, and to incorporate the state transformation in such a way to maintain the efficiency of event driven explicit simulation algorithms.

1. Introduction
With design trends continuously moving towards higher clock speed, higher density, and higher levels of integration on chip, the need for fast circuit simulation of large and complex integrated circuits is well known. There have been two major approaches in improving the efficiency of (circuit) transient simulation. Common to all approaches is the use of circuit partitioning and event driven simulation to exploit the spatial sparsity and the temporal latency of large circuits. One approach is to simplify circuit topology to a set of primitives, especially for MOS circuits, in addition to employing simplified device models, such as piecewise quadratic models, to derive the analytical solutions of the node voltages. This approach is embodied in the simulator ILLIADS [3]. The other major approach employs simplified device models to speed up the computation of the numerical solutions during a transient analysis. Piecewise constant (PWC) device models are employed in SPECS [4] and ADAPTS [5]. More general PWL models have been used in various timing analyzers and simulators [1][6]. The Adaptively Controlled Explicit Simulation (ACES) algorithm [2], has been developed to exploit the efficiency of explicit integration algorithm while ensuring the numerical stability of the integration at an incremental cost instead of the full cost of an implicit algorithm. The basic idea is to incrementally approach the steady state while ensuring the stability and efficiency of the integration algorithm.

In event driven simulation, an implicit assumption is that the states are independent so that they can be processed separately. If some of the states are strongly coupled or dependent, these states have to be processed as a group. In other words, the choice of the state variables for a state representation of a given circuit is not appropriate in the sense that the chosen states are not truly independent. This situation arises quite often in extracted circuits, which contain numerous parasitic resistors of small values. For example, given a resistor as shown in Fig. 1, the two capacitors at the two nodes of the resistor are usually chosen to be the state variables in a typical circuit formulation such as Modified Nodal Analysis (MNA). In the degenerate case of a zero valued resistor or voltage source, the two states are actually the same. Even if the value of the resistance is very small, i.e. the two nodes close to being shorted, then the two chosen state variables are not truly independent. In the following, a number of numerical problems associated with the case of small resistors mentioned above are discussed.

Matrix ill-conditioning
The first one is the numerical ill conditioning problem due to the resistor being stamped into the MNA circuit matrix equation as a conductance term with \( G = 1/R \) in the admittance form

\[
i_R = Gv_R = G(v_1 - v_2) .
\]

The reason that resistors are stamped as conductors is to avoid introducing extra unknowns into the MNA matrix equation, in which the capacitor voltages and inductor currents are chosen as state variables. However, if \( R \) is small, then \( G \) is large and it may degrade the numerical conditioning of the matrix equation. One way to avoid this problem is to stamp the resistor in its impedance form as

\[
v_R - Ri_R = v_1 - v_2 - R_i_R = 0
\]

Note that this impedance form introduces the extra unknown current variable \( i_R \) into the matrix equation. Note further that this stamping does not address the state dependency problem though it may alleviate the numerical conditioning problem.

State Dependency in Explicit Simulation
The state dependency problem is evident in explicit integration algorithm such as Forward Euler (FE). Given the Branch Constitutive Relation (BCR) of a capacitor as \( i_C = C\dot{v}_C \), the FE stamp of the capacitor at each time \( t \) is given by

\[
v_C(t) = v_C(t - \Delta t) + \frac{\Delta t}{C}i_C(t - \Delta t)
\]

where \( t - \Delta t \) is the previous time point and \( \Delta t \) is the time step. The above equation represents the BCR of a voltage source, and the
equivalent circuit of the simple structure shown in Fig. 1 using FE integration is shown in Fig. 2 below. As illustrated by Fig. 2, if \( R \) is small or zero in the degenerate case, the equivalent circuit includes an illegal loop of voltage sources. This state dependency problem will affect any explicit integration algorithm that simply update the capacitor voltages at a given time using only values at previous time points.

**State Dependency in Implicit Simulation**

The problem of state dependency is handled automatically in an implicit integration scheme because all the states are “implicitly” processed together at the same time. For example, using Backward Euler (BE) as the implicit integration scheme, the stencil of a capacitor in the MNA matrix equation at each time \( t \) can be written as

\[
v_C(t) = \frac{\Delta t}{C} i_C(t) + v_C(t - \Delta t) = G_{eq} i_C(t) + V_{eq}
\]

In circuit theoretic terms, this equation represents an equivalent circuit consisting of a resistor with conductance \( G_{eq} = (\Delta t)/C \) in series with a voltage source \( V_{eq} = v_C(t - \Delta t) \). Using these equivalent circuits for the two capacitors, the circuit shown in Fig. 1 can be redrawn as shown in Fig. 3 below. In this case, even if \( R = 0 \), the two equivalent voltage sources are still decoupled by the two equivalent conductors, and these voltage sources do not form an illegal loop.

The fundamental problem is then how to handle this state dependency problem without resorting to implicit integration algorithms for efficiency reasons. One simple solution is to preprocess a given circuit and delete from the circuit description all the resistors with small values below a specified tolerance. However, this approach may require setting different tolerances for different circuit configurations. Moreover, in some cases, it may be necessary to retain these small resistors in the circuit description. For example, given a long RC interconnect with many taps along the interconnect, it is necessary to retain the small resistors in each RC section of the interconnect to get the total resistance loading of the line. This problem obviates the need to develop an algorithmic approach, other than circuit pruning, to handle this problem. It is the goal of this paper to present such an approach using the event driven Adaptively Controlled Explicit Simulation (ACES) algorithm as an implementation vehicle. This paper is organized as follows. In section 2 of the paper, a simple example is presented to review the basic ACES algorithm as an implementation vehicle and illustrate the numerical problems due to state dependency. The main results of the paper are presented in section 3. This section discusses a general and systematic method to incorporate state transformation into event driven explicit simulation, especially its implementation in ACES. Section 4 presents some additional examples and simulation results on real circuits using state transformation. Section 5 summarizes the paper.

2. State Dependency in Event Driven Explicit Simulation

Consider a simple 2RC circuit as shown in Fig. 4. This simple example is used to review the basic ACES algorithm and to illustrate the problem of state dependence. The state equation of the circuit can be written as

\[
\begin{bmatrix}
C_1 & 0 \\
0 & C_2
\end{bmatrix}
\begin{bmatrix}
\dot{V}_1 \\
\dot{V}_2
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{R_1} + \frac{1}{R_2} & \frac{1}{R_2} \\
\frac{1}{R_2} & \frac{1}{R_2}
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2
\end{bmatrix}
+ \begin{bmatrix}
1 \\
0
\end{bmatrix} V_{in}
\]

(1)

where the capacitor voltages have been chosen to be the state variables. Let \( V_{in} \) be a unit step function, \( C_1 = C_2 = 1 \text{F} \), \( R_1 = 1 \Omega \), and \( R_2 = 10^{-6} \Omega \). Then the state equation can be written as

\[
\begin{bmatrix}
\dot{V}_1 \\
\dot{V}_2
\end{bmatrix}
= \begin{bmatrix}
-(1 + 10^6) & 10^6 \\
10^6 & -10^6
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2
\end{bmatrix}
+ \begin{bmatrix}
1 \\
0
\end{bmatrix} V_{in}
\]

(2)

Assume that the circuit is initially at rest, i.e., \( V_1 = V_2 = 0 \) at \( t = 0 \). Also assume a unit step input, i.e., \( V_{in} = 1 \) and \( V_{in} = 0 \) at \( t = 0^+ \). Then, applying the ACES algorithm to compute the time to quiescence [2] for the two states yields \( \Delta t_1 = -(\dot{V}_1(0))/\dot{V}_1(0) \approx 10^{-6} \) and \( \Delta t_2 = 0 \), which implies that the input excitation has not reached the second node at this time point. In other words, one is interested only in non-zero time step at each time point. Hence, \( \Delta t = \Delta t_1 = 10^{-6} \).

Continue the ACES algorithm to the next time point, \( t_1 = t_0 + \Delta t = 10^{-6} \). Note that \( V_1 \) is now in quiescence and it should not dictate the computation of the next time step. Hence the time to quiescence for \( V_2 \) is given by

\[
\Delta t = -(\dot{V}_2(t_1))/\dot{V}_2(t_1) = 1 + 10^{-6}.
\]

Then, at the next time point, \( t_2 = t_1 + \Delta t \), both states will be quiescent and the simulation is completed. The result of this simple simulation for \( V_2 \) is shown in Fig. 5 together with the result obtained by AS/X, the IBM internal circuit simulator. Fig. 5 shows the result obtained by ACES with a voltage error tolerance \( \varepsilon_v = 0.001 \), where
be written as space of the circuit. Compute the eigen-decomposition of the system.

This discrepancy illustrates the problem in assuming the independence of strongly coupled state variables.

\[ \tau_R \cdot \varepsilon_2 \]

shows significant discrepancy as compared to the AS/X result. This is due to the fact that the two states are treated as independent and processed one at a time.

From the circuit point of view, when \( v_1 \) enters quiescence, ACES replaces \( C_1 \) by a zero valued current source, i.e. \( v_1 \) becomes an independent and processed one at a time.

In order to resolve this problem, an appropriate state space representation of the circuit must be chosen in such a way that the states of circuits with large linear subcircuits. The advantage of this approach is exactly the goal of numerical integration algorithm. So the question is how to determine an appropriate state transformation. One obvious choice is the eigen-space representation is an ideal space for event driven explicit simulation because the modes of the circuit are decoupled and can be processed sequentially. However, this approach is clearly impractical due to a number of reasons. First the computation of the eigen-decomposition is expensive. Moreover, once the eigen-decomposition is known, the solution can be computed directly without resorting to numerical integration. Avoiding the eigen-decomposition is exactly the goal of numerical integration algorithm. So the question is how to determine an appropriate state transformation. One possible approach is to perform an approximate or exact eigen-computation. This alleviates the cost of a full eigen-computation. However, for non-linear circuits, an eigen-computation needs to be performed for every new linearization of the nonlinear equations. Another approach is to perform a partial eigen-computation. This is a basically a model reduction problem, especially for large linear circuits. This approach is a standard technique for reduced order modeling of large linear circuits in general nonlinear simulation of circuits with large linear subcircuits. The advantage of this approach is that it can prune out the non dominant eigen-states, such as the state due to the large eigenvalue \( (2(10^6)) \) in the 2RC example above. However, this approach is not practical for nonlinear circuits.

In general, there is a need to have a simple transformation matrix in order not to degrade the efficiency of event driven explicit simulation technique such as ACES. Note again that one key assumption of circuit level (not logic level or macromodel or block level) event driven simulation is that the states are independent or DC decoupled so that the excitation can be propagated from the input source to the various nodes in the circuit in an event driven manner. For example, the unit step excitation from the input source of the above 2RC circuit will propagate first to node 1, and then to node 2. However, when the two nodes are strongly DC coupled, the algorithm will yield erroneous results as discussed in the previous section. The key idea is to process the two states simultaneously. Note that the eigen-transformation described in the previous section transforms the input matrix

\[ \Delta t = \min(\Delta t_1, \Delta t_2) = \min\left(\frac{y_1}{\hat{y}_1}, \frac{\hat{y}_2}{\hat{y}_2}\right) = \frac{1}{2}(10^{-6}) \]

At the next time point \( t_1 = \Delta t = 0.5(10^{-6}) \), \( \hat{y}_1 = \hat{y}_2 = 0 \). Hence \( y_1 \) will stay in quiescence and the time to quiescence for \( y_2 \) can be computed as

\[ \Delta t = \frac{\hat{y}_2}{\hat{y}_2} = 0.5 \]

In this case, the ACES algorithm captures the correct time constant of the original circuit. This simple example has clearly illustrated the need to select an appropriate state space in which the state variables are truly independent so that an event driven explicit algorithm such as ACES can be applied to provide accurate simulation results.

3. State Transformation for Event Driven Explicit Simulation

The eigen-space representation is an ideal space for event driven explicit simulation because the modes of the circuit are decoupled and can be processed sequentially. However, this approach is clearly impractical due to a number of reasons. First the computation of the eigen-decomposition is expensive. Moreover, once the eigen-decomposition is known, the solution can be computed directly without resorting to numerical integration. Avoiding the eigen-decomposition is exactly the goal of numerical integration algorithm. So the question is how to determine an appropriate state transformation. One possible approach is to perform an approximate or exact eigen-computation. This alleviates the cost of a full eigen-computation. However, for non-linear circuits, an eigen-computation needs to be performed for every new linearization of the nonlinear equations. Another approach is to perform a partial eigen-computation. This is a basically a model reduction problem, especially for large linear circuits. This approach is a standard technique for reduced order modeling of large linear circuits in general nonlinear simulation of circuits with large linear subcircuits. The advantage of this approach is that it can prune out the non dominant eigen-states, such as the state due to the large eigenvalue \( (2(10^6)) \) in the 2RC example above. However, this approach is not practical for nonlinear circuits.

In general, there is a need to have a simple transformation matrix in order not to degrade the efficiency of event driven explicit simulation technique such as ACES. Note again that one key assumption of circuit level (not logic level or macromodel or block level) event driven simulation is that the states are independent or DC decoupled so that the excitation can be propagated from the input source to the various nodes in the circuit in an event driven manner. For example, the unit step excitation from the input source of the above 2RC circuit will propagate first to node 1, and then to node 2. However, when the two nodes are strongly DC coupled, the algorithm will yield erroneous results as discussed in the previous section. The key idea is to process the two states simultaneously. Note that the eigen-transformation described in the previous section transforms the input matrix

\[ A = \begin{bmatrix} \frac{1}{R_1} + \frac{1}{R_2} & \frac{1}{R_2} \\ \frac{1}{R_2} & -\frac{1}{R_2} \end{bmatrix} \]

where

\[ T = \begin{bmatrix} 1 & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 1 & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \]

Define the transformed state variables as \( \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = T^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \). The transformed state equation can then be written as

\[ \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \Lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + T \begin{bmatrix} 1/V_{in} \\ 0 \end{bmatrix} = \Lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} V_{in} \]

Now apply the ACES algorithm to this system. At \( t = 0 \), \( y_1 = y_2 = 0 \). From the state equations, it can be seen that \( \dot{y}_1 \neq 0, \dot{y}_2 \neq 0, \dot{y}_1 = 0, \) and \( \dot{y}_2 \neq 0 \). Hence the time step can be computed as

\[ \Delta t = \min(\Delta t_1, \Delta t_2) = \min\left(\frac{y_1}{\hat{y}_1}, \frac{\hat{y}_2}{\hat{y}_2}\right) = \frac{1}{2}(10^{-6}) \]

At the next time point \( t_1 = \Delta t = 0.5(10^{-6}) \), \( \hat{y}_1 = \hat{y}_2 = 0 \). Hence \( y_1 \) will stay in quiescence and the time to quiescence for \( y_2 \) can be computed as

\[ \Delta t = \frac{\hat{y}_2}{\hat{y}_2} = 0.5 \]

In this case, the ACES algorithm captures the correct time constant of the original circuit. This simple example has clearly illustrated the need to select an appropriate state space in which the state variables are truly independent so that an event driven explicit algorithm such as ACES can be applied to provide accurate simulation results.

3. State Transformation for Event Driven Explicit Simulation

The eigen-space representation is an ideal space for event driven explicit simulation because the modes of the circuit are decoupled and can be processed sequentially. However, this approach is clearly impractical due to a number of reasons. First the computation of the eigen-decomposition is expensive. Moreover, once the eigen-decomposition is known, the solution can be computed directly without resorting to numerical integration. Avoiding the eigen-decomposition is exactly the goal of numerical integration algorithm. So the question is how to determine an appropriate state transformation. One possible approach is to perform an approximate or exact eigen-computation. This alleviates the cost of a full eigen-computation. However, for non-linear circuits, an eigen-computation needs to be performed for every new linearization of the nonlinear equations. Another approach is to perform a partial eigen-computation. This is a basically a model reduction problem, especially for large linear circuits. This approach is a standard technique for reduced order modeling of large linear circuits in general nonlinear simulation of circuits with large linear subcircuits. The advantage of this approach is that it can prune out the non dominant eigen-states, such as the state due to the large eigenvalue \( (2(10^6)) \) in the 2RC example above. However, this approach is not practical for nonlinear circuits.

In general, there is a need to have a simple transformation matrix in order not to degrade the efficiency of event driven explicit simulation technique such as ACES. Note again that one key assumption of circuit level (not logic level or macromodel or block level) event driven simulation is that the states are independent or DC decoupled so that the excitation can be propagated from the input source to the various nodes in the circuit in an event driven manner. For example, the unit step excitation from the input source of the above 2RC circuit will propagate first to node 1, and then to node 2. However, when the two nodes are strongly DC coupled, the algorithm will yield erroneous results as discussed in the previous section. The key idea is to process the two states simultaneously. Note that the eigen-transformation described in the previous section transforms the input matrix
In the original state space, the input directly affects only $v_1$. In the transformed state space, the input directly affects both $y_1$ and $y_2$ simultaneously. Hence the key idea in constructing a simple transformation matrix is to make sure that all the transformed states will be dependent on all the original states. In this way, any input that affects at least one state in the original space will affect all the states in the transformed space. In this way, the set of coupled transformed states can be processed simultaneously during the “electrical” event propagation. For the 2RC circuit example, one such simple transformation matrix is given as

$$T = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$ (6)

This matrix looks like a simple scaling of the transformation matrix $T^{-1}$ (the inverse eigenvector matrix) discussed in the previous section. In general, this is not the case. The transformation matrix in Eq. (6) is simply to make every transformed states dependent upon all the original state variables. In order to bring out the effect of the capacitor charges, let’s choose $C_1 = 0.5$ and $C_2 = 2$. Also select the capacitor charges, denoted by $q_1$ and $q_2$ as the state variables in the original space, i.e., $[q_1 \; q_2]^T = [C_1 v_1 \; C_2 v_2]^T$. And the transformed state variables are defined as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = T \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

An ACES simulation is then performed for this circuit. Note that the times to quiescence (an “electrical” event in ACES) are computed for the transformed state variables. The local truncation error control is performed for the original state variables (the node voltages) in this case. The waveform at node 2 as obtained by the ACES simulation with state transformation is shown in Fig. 6 together with the AS/X simulation result. Note that the state transformation helps ensure the accuracy of ACES simulation.

In general, if there are no coupling among the states, then the transformation matrix is simply an identity matrix. Suppose there are $n$ coupled states, then the submatrix corresponding to these $n$ states will be of the form

$$T_{nxn} = \begin{bmatrix} +1 & -1 & -1 & \ldots & -1 \\ +1 & +1 & -1 & \ldots & -1 \\ +1 & +1 & +1 & \ldots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ +1 & +1 & +1 & \ldots & +1 \end{bmatrix}$$

For example, given a circuit with 7 states among which there are two sets of coupled states: {2,3,4} and {6,7}, then the transformation matrix for this circuit can be written as

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

This simple structure allows the transformation matrix to be constructed in a systematic manner during the processing of the circuit. For example, a simple two phase procedure to construct this type of transformation matrix can be described as follows. First initialize the transformation matrix $T$ to be an identity matrix. Then in the first phase of processing the circuit, check if the terminals (which are assumed to be states for simplicity) or nodes of a given resistive element (or device) are coupled. If the nodes are coupled, then enter the appropriate nonzero entries, which corresponds to the indices of the nodes in the state vector, in the transformation matrix. The purpose of this phase is to record the coupling information in the matrix. In the second phase, the transformation matrix will be processed row by row to determine different groups of coupled states. Then the submatrices of the type described above for these groups of coupled states will be recorded in the transformation matrix.

The next question is how to incorporate state transformation into ACES. A straightforward approach is simply transform the original state equation into a new state equation and perform the analysis in the transformed space. However, it should be noted that the transformed space is used to compute the time to quiescence for the transformed states while the local truncation error is still performed in the original space. Therefore, it is simpler to impose the constitution relation or the quiescence condition for the transformed states in terms of the original states to solve for the necessary variables in the original space. Then the variables in the transformed space can be computed by applying the transformation matrix to the original states. These ideas will be illustrated below using a simple Modified Nodal Analysis (MNA) circuit formulation in the derivative space [2].

In order to simplify the presentation and to bring out the difference between the basic ACES algorithm and the modified algorithm to handle state transformation, assume that there are no inductors,
current sources, floating capacitors, and floating voltage sources. These circuit elements can be incorporated into the formulation in a straightforward manner once the basic algorithm has been established. Under these assumptions, during a quiescence period [2], the MNA formulation of the circuit in the derivative space can be written as

$$
\begin{bmatrix}
G_{cc} & G_{cv} & E_{cc} & 0 \\
G_{vc} & G_{vv} & 0 & E_v \\
K_c & 0 & E_{qc} & 0 \\
0 & U_v & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\dot{v}_c \\
\dot{v}_v \\
\dot{q}_c \\
\dot{u}_v
\end{bmatrix}
=

\begin{bmatrix}
0 \\
0 \\
h_c \\
0
\end{bmatrix}
$$

where $G_{cc}$, $G_{cv}$, $G_{vc}$, and $G_{vv}$ are submatrices of the conductance matrix formed by resistive elements; $E_{cc}$, $E_{qc}$, $E_v$, and $U_v$ are submatrices of 1’s and 0’s to pick out the appropriate capacitor/source voltages or currents; $K_c$ and $h_c$ represent the changeable portion of the MNA equation during the simulation depending upon whether the states are in quiescence or not. In general, the first two rows of Eq. (7) represents Kirchhoff Current Law (KCL) equations at all voltage and capacitance nodes. The third row represents the Branch Constitutive Relations (BCR) and/or the quiescence conditions for the capacitors. The fourth row represents the BCRs for the voltage sources.

Now define the state transformation $y_c = Tq_c$. Initially when all the states are non-quiescent, the BCRs for the transformed states can be written as $\dot{y}_c = TC\dot{v}_c$. In circuit theoretic terms, the transformed states can be considered as voltage controlled current sources. In this case, the stamps in the MNA matrix of Eq.(7) can be set as follows:

$K_c = TC$, $E_{qc} = 0$, and $h_c = \dot{y}_c$. Suppose at time $t$, the $j$th (transformed) state enters quiescence. Then $\dot{y}_{cj}(t) = 0$ and the stamps in the MNA matrix can be modified as $k_{cj}^T = 0$, $e_{qc,j}^T = 0$, and $h_{cj} = 0$, where $k_{cj}^T$ is the $j$th row of $T$, $k_{cj}^T$ is the $j$th row of $E_{qc}$, and $e_{qc,j}^T$ is the $j$th row of $E_{qc}$. In this case, the transformed state in quiescence can be treated as a zero valued current controlled current source. In this case, partition the vector of transformed states as

$$
y_c = \begin{bmatrix}
y_{cn} \\
y_{cq}
\end{bmatrix} = \begin{bmatrix} T_n & T_s \end{bmatrix} q_c
$$

where $y_{cn}$ denotes the subvector of transformed states that are not in quiescence, and $y_{cq}$ denotes the subvector of transformed states that are in quiescence; $T_n$ and $T_s$ are appropriate partitions of the transformation matrix $T$. Then the MNA equation as shown in Eq.(7) can be rewritten as

$$
\begin{bmatrix}
G_{cc} & G_{cv} & E_{cc} & 0 \\
G_{vc} & G_{vv} & 0 & E_v \\
T_n C & 0 & 0 & 0 \\
0 & 0 & T_s & 0
\end{bmatrix}
\begin{bmatrix}
\dot{v}_c \\
\dot{v}_v \\
\dot{q}_c \\
\dot{u}_v
\end{bmatrix}
=

\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}
$$

where the third row of Eq.(9) describes the non-quiescent transformed states, and the fourth row describes the quiescent transformed states. The modified ACES algorithm with state transformation is summarized in the pseudocode below.

**Algorithm with state transformation:**

1. **Update sources**

2. **Update original state variables (for all states):**

   $$
   \dot{q}_c(t) = \dot{q}_c(t - \Delta t) + \dot{q}_c(t - \Delta t)\Delta t
   $$

   $$
   \dot{v}_c(t) = \dot{v}_c(t - \Delta t) + \dot{v}_c(t - \Delta t)\Delta t
   $$

3. **Check quiescence conditions for $y_c$**

4. **Setup the MNA matrix and solve for $\dot{y}_c(t^+)$, $\dot{y}_c(t)$, and $\dot{q}_c(t)$**

5. **Compute $\dot{q}_c(t^+)$ and $\dot{v}_c(t) = C^{-1}\dot{q}_c(t)$ for all states**

6. **Compute time to quiescence for non-quiescent states, $t_{qs} = -[\dot{y}_c(t)/\dot{y}_c(t)]$**

7. **Set $\Delta t = \text{min}(t_{qs})$**

   In the above algorithm, the quiescence conditions are checked for the transformed states, not the original states. The advantage of this algorithm is that there is no need to explicitly construct the transformed space. The solution procedure is still performed in the original space with minimal modification of the matrix equation to impose the BCRs and/or the quiescence conditions for the transformed states. In the next section, some examples will be presented to assess the overhead of the modified ACES algorithm and its accuracy as compared with the basic ACES algorithm.

4. **Examples**

The first example is a simple nonlinear circuit consisting of a driver driving four buffers. Each connection between the driver and a given buffer is an 8 RC interconnect model. The schematic of the circuit is shown in Fig. 7. This example is to illustrate the complication due to circuit partitioning which may couple a port to an internal state of a given partition. This situation arises when parasitic resistances are added to the terminals of a MOSFET model. For example, there are five partitions P1 to P5 in the circuit shown in Fig. 7. Ports numbered 1 to 3 will be coupled to the internal states of partition P1 if the resistor at the end of the lines T1 to T3 are shorted. In a standard ACES algorithm, a partition is processed only if a port attached to the partition has an event. A port event is initiated when the volt-
age slope at the port has an appreciable change. Otherwise, the partition is not processed even if the port information is updated when another partition connected to the same port has an event. For example, assume partition P2 has an event at a given time point. In processing partition P2, the voltage and the slope at port 1 will be updated. If the slope has a significant change (above a given tolerance), then partition P1 will be processed at the same event time. Otherwise, P1 will not be processed. However, in the case port 1 is coupled to an internal state of P1, P1 will have to be processed regardless of whether P1 has a significant change in voltage slope or not. This processing is necessary to take the coupling into account when computing all the necessary information for the next event. A sample run is performed for this circuit with the first, last, and two middle resistors of the lines are shorted to produce different modes of coupling among the internal states and the ports of P1. In this case, the ACES simulation without state transformation requires 309718 events and 34.36 CPU seconds. The ACES simulation with state transformation requires only 3822 events and 2.77 CPU seconds. The ACES simulation results are compared with AS/X simulations and shown in Fig. 8 at the output of partition P2.

Fig. 8: ACES simulation results (with and without state transformation) as compared to AS/X at the output of partition P2 as shown in Fig. 7.

The last example is an industrial design of a 4 bit ALU. The circuit consists of 282 MOSFETs and 156 nodes. Without parasitic resistances added to the MOSFET models, the ACES simulation requires only 7298 events and 1.17 CPU seconds while the AS/X simulation requires 50 CPU seconds. With a number of parasitic resistances added to the circuit, the AS/X simulation requires about the same 50 CPU seconds. In this case, the ACES simulation with state transformation requires 17640 events and 5.73 CPU seconds. The waveforms at an output node of the circuit as obtained by ACES simulation with and without parasitic resistances and by AS/X are shown in Fig. 9. Note that the difference between the ACES waveforms is due to the state transformation. In the case of no parasitic resistances (and no transformation), the time to quiescence is computed for the original state. In the case of state transformation, the time to quiescence is computed for the transformed states.

5. Summary

This paper has identified the dependency of the states in the state space description of a given circuit as the cause for some numerical problems in event driven explicit simulation algorithms such as ACES. State transformation has also been proposed to address this problem. A novel algorithm has been developed to derive systematically a simple state transformation and to include such a transformation in an efficient manner. A number of examples have been included to illustrate the concept and the complication in using state transformation as well as to demonstrate its effectiveness.

References